Mestrado em Informática e Sistemas

Waldo, the Virtual & Intelligent Cyber Analyst

Relatório de Estágio apresentado para a obtenção do grau de Mestre em Informática e Sistemas
Especialização em Desenvolvimento de Software

Autor
Daniel Ribeiro Margarido

Orientador
Francisco José Pereira
Professor do Departamento de Engenharia Informática e Sistemas
Instituto Superior de Engenharia de Coimbra

Supervisor
Rui Gonçalo Amaro
Dognædis

Coimbra, Setembro, 2017

Resumo

O objetivo deste estágio é desenvolver uma plataforma inteligente, que constitua um analista de segurança, através de agregação e correlação de várias fontes, apresentado-as ao operacional de segurança através de uma aplicação web. Este objetivo foi atingido com a implementação de um sistema que: Aplica os cálculos das redes bayesianas para entender o correlacionamento de eventos de vários IDS, utiliza abordagens que conseguem encontrar comportamentos suspeitos a partir da análise da variação destes correlacionamentos e emprega raciocínio baseado em casos para comparar esses comportamentos com outros registados anteriormente. Ao testar o sistema na rede da empresa, foram detetados alguns comportamentos suspeitos, que normalmente não seriam detetados pelos operadores com a sua forma de análise regular. Este sistema mostrou ser bastante importante, não para substituir os operadores, mas para servir de apoio à análise da elevada quantidade de dados que estes recebem. Ao mesmo tempo, fornece um tipo de análise diferente da dos operadores, podendo detetar problemas que não são percebidos através do método normal.

Abstract

The objective of this internship is to develop an intelligent platform, which is a security analyst, through aggregation and correlation of several sources, presenting them to the security operative on a web application. This objective was achieved with the implementation of a system that: Applies calculations of Bayesian networks to understand the correlation between the various IDS events, uses approaches that are able to find suspicious behaviour from the analysis of variation of these correlations and employs case-based reasoning to compare these behaviours with those previously reported. While testing the system in the organization network, some suspicious behaviours were detected, which normally wouldn't be detected by operators through their regular analysis. This system proved to be important, not to replace the operatives, but to serve as support in the analysis of the high volume of data they receive. At the same time, it supplies a new type of analysis, which can detect problems that wouldn't normally be perceived.

Agradecimentos:
Agradeço ao Professor Francisco José Pereira por todo o apoio e dedicação que me deu enquanto meu orientador.
Aos Engenheiros Rui Gonçalo Amaro e André Manuel Pinheiro por todo o apoio, dedicação, paciência e amizade que demonstraram ao longo de todo o percurso deste estágio.
A toda a equipa da *Dognædis*, pelo fantástico acolhimento, por toda a ajuda disponibilizada, conhecimentos transmitidos, condições de trabalho e auxílio precioso na revisão deste documento.
Agradeço ainda ao todos os docentes do ISEC que de alguma forma ajudaram na minha formação enquanto aluno da LEI e do MIS.

Dedicatória:
Dedico este trabalho à minha família, aos meus amigos e à *Dognædis*, em especial aos meus pais Amândio Margarido e Clarinda Margarido, ao meu irmão Rui, às minhas irmãs Mónica e Maria, e à minha namorada Solange, pelo apoio, paciência e compreensão em todos os momentos.

Conteúdo

Lista de Figuras

Lista de Tabelas

Glossário

Acrónimo/Sinónimo	Significado
0-Day	É uma vulnerabilidade não pública que pode ser explorada para afetar programas, dados, sistemas ou redes.
Ajax	Asynchronous JavaScript and XML, é um conjunto de técnicas de desenvolvimento web utilizadas do lado do cliente para enviar e receber dados de um servidor de forma assíncrona, não interferindo com a visualização e comportamento da página atual.
Broker	Padrão de desenho de software que serve para esconder os detalhes de implementação de comunicação com um serviço externo ao sistema, encapsulando-os num componente separado do que contém a lógica de negócio[7].
CBR	Case-based Reasoning. Sistema que se baseia em casos e conhecimento passado, em vez de regras, para tentar oferecer soluções para novos casos.
Cluster	Define-se por um grupo de elementos semelhantes ou bastante próximos.
Constituency	Refere-se à separação de entidades responsáveis por cada rede/-cliente.
Coverage	É uma medida utilizada para descrever a quantidade de código fonte de um programa que é executado quando um conjunto particular de testes é executado.
Cross-platform	Pode ser utilizado em pelo menos duas plataformas diferentes, por exemplo GNU/Linux, Mac OS X, Windows, entre outros.
Cross-Validation	Técnica utilizada para criar um conjunto maior de dados para treino e teste a partir de um dataset mais pequeno [8].
CSV	Comma Separated Values, é um formato de texto que representa dados em tabelas, estando os valores em linhas e separados por virgulas.
DDOS	Distributed Denial Of Service ou Ataque Distribuído de Negação do Serviço.
EARS	A abordagem simples à sintaxe de requisitos(The Easy Approach to the Requirements Syntax[9]).
Error	Um estado interno errado tendo como origem uma Fault [10].

Acrónimo/Sinónimo	Significado
Failure	Comportamento externo incorreto relativo aos requisitos definidos ou outro comportamento esperado [10].
Fault	Um defeito estático no software [10].
Feature	Esta nomenclatura vem das áreas de machine learning e estatística, e associa-se a atributos de um objecto, como "cor" e "textura" que poderiam ser features do objeto laranja.
Framework	É um conjunto de conceitos utilizados para resolver um problema de um domínio especifico, normalmente possuem funcionalidades genéricas e comuns o que permite a sua utilização em diferentes aplicações.
GNU	GNU's Not Unix, é um sistema operativo começado por Richard Stallman e construido pela comunidade de software livre com o objetivo de construir um sistema operativo totalmente livre[11].
GNU/Linux	Sistema regularmente referido como "Linux", sendo este a combinação das ferramentas e programas do GNU juntamente com o Kernel Linux, criando um sistema operativo completo[11].
HTML	HyperText Markup Language, é uma linguagem utilizada para estruturar páginas web.
IDE	É um *Integrated Development Environment*, oferecendo várias ferramentas integradas que ajudam ao desenvolvimento de software.
IDS	Sistema de deteção de intrusões que monitoriza uma rede ou um sistema de modo a encontrar atividade maliciosa ou quebras de políticas de segurança.
JSON	Javascript Object Notation, é um formato, que utiliza texto estruturado de modo a ser facilmente compreensível para humanos, para transmitir dados de objetos e ao mesmo tempo é uma estrutura simples para as máquinas criarem e analisarem.
Label	Label provem das áreas de machine learning e estatística, descrevendo uma classe de objeto. Por exemplo na classificação de várias frutas poderíamos ter como labels "laranja" e "maça".
Linux	Kernel construido por Linus Torvalds e a comunidade open-source que foi licenciado com uma licença livre[11].
Malware	É software malicioso destinado a infiltrar-se num sistema alheio de forma ilícita, com o intuito de causar danos, alterações ou roubo de informações.

Acrónimo/Sinónimo	Significado
ORM	Object-Relational Mapping, é uma técnica utilizada para criar uma base de dados virtual representada pelas classes e objetos a partir de uma linguagem de programação. Permite a realização de operações à base de dados de forma transparente apenas alterando os valores dos objetos e permite actualizar a estrutura da base de dados com modificações à estrutura das classes.
Overfitting	Designa quando um modelo de machine learning perde a sua capacidade de generalização e se torna altamente reativo a pequenas variações nos dados [12].
Plug-in	Programa criado para adicionar uma função a um programa maior, criando uma funcionalidade especial ou muito especifica.
Portolan	Plataforma de inteligência para segurança desenvolvida pela Dognædis.
POST	É um tipo de pedido suportado pelo protocolo HTTP. Este é utilizado para enviar dados para um servidor web.
Precision	Refere-se à percentagem de elementos relevantes do conjunto de elementos selecionados[13].
Recall	É a percentagem de elementos relevantes do conjunto de todos os elementos relevantes existentes[13].
Rootcheck	Verificar se houveram acessos como root[14].
Security Orchestration	É um método de interligar várias ferramentas e sistemas dispares, sendo a camada de ligação que alinha procedimentos e permite automatização da segurança [15].
Stakeholder	Uma pessoa, grupo ou organização que tenha interesses em uma organização.
URI	Uniform Resource Identifier, é uma string de caracteres utilizada para identificar um recurso. Sendo por exemplo o URL um tipo de URI.
URL	Uniform Resouce Locator ou web address, é a referência para um recurso web especificando a sua localização numa rede e um mecanismo para buscar o recurso.
XML	Extensible Markup Language, esta define várias regras para codificar documentos de modo a poderem ser lidos por humanos e por máquinas.

Acrónimo/Sinónimo	Significado
Trojan	Um trojan é uma aplicação maliciosa que aparenta fazer uma tarefa mas na verdade faz outra.
Trojan backdoor	Um trojan backdoor difere do trojan no ponto que também abre uma porta escondida para o sistema, são também chamados de RAT(Remote Access Trojan).

Capítulo 1

Introdução

Esta dissertação descreve o trabalho desenvolvido no estágio integrado no Mestrado em Informática e Sistemas(MIS) do ramo de Desenvolvimento de Software do Departamento de Engenharia Informática e Sistemas do Instituto Superior de Engenharia de Coimbra no ano letivo 2016/2017. O estágio tem a duração de 1200 horas, foi realizado na empresa Dognædis, tendo como objetivo a especificação e implementação do projeto *Waldo, the Virtual & Intelligent Cyber Analyst*. Este é um sistema inteligente, capaz de evoluir ao longo do tempo, que realiza classificação de comportamentos suspeitos na rede através do correlacionamento de alertas de segurança.
Neste capítulo é apresentada a entidade acolhedora, a empresa Dognædis, é feito um enquadramento do projeto, são indicados os seus principais objetivos e a estrutura desta dissertação.

1.1 Entidades Acolhedoras

A entidade acolhedora deste estágio foi a Dognædis[16], uma empresa focada em segurança das tecnologias de informação.
A Dognædis foi fundada por quatro engenheiros da Universidade de Coimbra: Francisco Rente, Hugo Trovão, Mário Zenha Rela, e Sérgio Alves, mas teve também na sua génese, uma equipa de investigadores da Universidade de Coimbra. Esta equipa esteve previamente na origem de uma CSIRT(Computer Security Incident Response Team), o CERT-IPN, alojado no Instituto Pedro Nunes, uma interface de transferência de tecnologia criada pela Universidade de Coimbra, em Portugal. Após cinco anos de atividade, e devido ao aumento de sucesso e feedback positivo de diversas organizações privadas e governamentais, a Dognædis lançou-se como uma companhia privada.
Entre outros, a Dognædis desenvolveu o *CodeV*[18], produto premiado pelo BES Concurso Nacional De Inovação. Este apresenta-se como um software inteligente que tem a capacidade de detetar problemas de segurança no soft-

ware de terceiros, em todas as suas fases de desenvolvimento.
A Dognædis assume, acima de tudo, uma posição de luta face à atual conjuntura de crise, vendo o futuro da economia portuguesa além fronteiras.
O ISEC (Instituto Superior de Engenharia de Coimbra)[17], segue a tradição do ensino prático associado à engenharia bem como a forte ligação ao tecido empresarial e industrial são imagens de marca do ISEC. Este tem como visão institucional ser uma referência de excelência no ensino, reconhecido nacional e internacionalmente por serviços de qualidade e relevância social, com práticas flexíveis, criativas e inovadoras. Pretende ainda ser um parceiro privilegiado das organizações empresariais e das famílias da região onde se insere pela orientação maioritáriamente prática, fundada num rigoroso conhecimento teórico, que imprime a todas as suas actividades.

1.2 Âmbito

A monitorização contínua e resposta a incidentes que possam ocorrer sobre as redes de comunicação e serviços de informação cria a necessidade de analistas de segurança que estejam constantemente a correlacionar informação patente nestes registos e plataformas para detetar possíveis incidentes de segurança e conseguir que estes sejam tratados e mitigados.

O objetivo deste trabalho é o desenvolvimento de uma plataforma de aprendizagem, que permita uma ampla simplificação do trabalho manual de correlação de eventos. Esta aplicação computacional permitirá melhorar a eficiência do operador de segurança, na reação a possíveis eventos de intrusão e a aprendizagem continua entre operador e plataforma. Tal facilita a compreensão que determinados registos possam ter para a identificação de incidentes.

Antes do inicio do estágio a monitorização contínua e a resposta a incidentes era feita de forma manual e tinha as seguintes características:

- Todos os eventos ligeiramente mais suspeitos tinham de ser analisados;
- Para cada evento suspeito era necessário ver os logs que foram gerados. De forma a perceber se é ou não uma intrusão é preciso efetuar pesquisa e análise de vários tipos de dados em bruto que estejam relacionados;
- Existem muitos dados que possam ser analisados, para conseguir uma boa análise poderia ser necessário bastante tempo de pesquisa;
- Muitos dos alertas derivam de problemas já conhecidos, sendo que na sua maioria não apresentam grande perigo. No entanto, esse mesmo tipo de alerta esporadicamente pode apresentar uma intrusão. Desta forma, o operador tinha de realizar sempre a análise mesmo que em 99% dos casos não haja nenhum problema real.

1.3 Objetivos

Face ao modo de funcionamento descrito na secção anterior e às necessidades levantadas pela empresa na proposta de estágio presente no apêndice D, pretendia-se desenvolver uma solução que permitisse:

- Desenvolver um sistema inteligente que consiga de forma autónoma perceber comportamentos estranhos nas redes dos clientes e classificar os mesmos;
- Implementar um mecanismo de aprendizagem através do qual a plataforma detete e registe os comportamentos dos operadores de monitorização, no seu trabalho do dia a dia;
- Aprender com tais comportamentos, levando a que caso a mesma situação se repita, sugira ao operador o conjunto de comportamentos que tenham sido tomados anteriormente;
- Moldar a aprendizagem da plataforma enviando feedback, de modo a determinar se a informação correlacionada, é ou não útil, para a deteção/seguimento de determinado incidente;
- Sugerir nova informação ao operador, com base em comportamentos anteriores. Desta maneira mesmo que esta informação não tenha sido relevante nos casos anteriores esta ainda se pode mostrar relevante no âmbito da análise em que se encontra.

Durante o processo de compreensão do domínio do problema e do levantamento da Análise de Requisitos, descrita no capítulo 2, entendeu-se que os objetivos não seriam suficientes. Após alguma análise e discussão os objetivos foram reformulados:

- Detetar incidentes de modo autónomo através da correlação de eventos de segurança de várias fontes;
- Apresentar investigações de incidentes a operadores, demonstrando as correlações de informação feitas pelo Waldo e a informação que possa ser pertinente analisar;
- Moldar ao longo do tempo a aprendizagem da plataforma, enviando feedback sobre a informação correlacionada, de modo a entender se esta é útil para a avaliação de determinado incidente;
- Comunicar com Portolan, uma plataforma de Cyber Intelligence da organização, para obter informações extra que possam ajudar nas investigações.

1.4 Contribuição

Quando o analista Waldo estiver em produção, as características anteriores vão ser simplificadas e melhoradas nos seguintes aspetos:

- Todos os eventos de segurança continuam a ter de ser analisados, contudo o Waldo pode avisar o operador antes deste se aperceber do evento, apresentando logo a sua investigação já com os dados correlacionados. Neste caso o operador terá apenas de confirmar ao Waldo que o problema foi encontrado;
- Quando apresentar uma investigação, mesmo esta não sendo uma intrusão, já oferece ao operador os dados correlacionados com os eventos de segurança que encontrou, facilitando o tempo e a qualidade da pesquisa mesmo com dados em bruto;
- Pode ser ensinado a reconhecer padrões de forma eficaz. De modo a apresentar uma investigação quando houverem dados novos em relação ao mesmo alerta que na maioria das vezes não representava problema algum.
- A existência de um analista virtual no meio destas fontes de informação possibilita a expansão da quantidade de fontes e de áreas de análise. Desta forma, iremos melhorar tanto a qualidade das previsões como a ajuda ao operador que com a sua limitação humana naturalmente teria problemas em conseguir absorver tanta informação num curto espaço de tempo.

1.5 Estrutura do Documento

Os capítulos seguintes estão organizados da seguinte forma:

- **Análise de Requisitos** - O segundo capítulo dá a conhecer os requisitos funcionais e os atributos de qualidade levantados para a aplicação.
- **Estado da Arte** - O terceiro capítulo procura estudar ferramentas de propósito semelhante àquela que o autor se propõe a desenvolver, analisando-as e comparando-as com a solução proposta. Visa também a análise de artigos que possam sugerir diferentes abordagens e conteúdo pertinente para o planeamento do sistema;
- **Planeamento** - No quarto capítulo são ponderados os riscos que poderiam surgir durante o projeto e é apresentado o planeamento da atividade a realizar;

- **Arquitetura do Sistema** - O quinto capítulo especifica o enquadramento do projeto a desenvolver na infraestrutura do sistema já existente, a arquitetura proposta para a implementação dos requisitos levantados e quais as ferramentas adotadas para desenvolvimento da solução;
- **Metodologia de Desenvolvimento** - O sexto capítulo indica a metodologia escolhida pelo autor para o lidar com o desenvolver do projeto;
- **Implementação** - O sétimo capítulo procura apresentar em detalhe as decisões de implementação referentes a cada um dos requisitos definidos na especificação arquitetural;
- **Verificação e Validação** - O oitavo capítulo evidencia os testes de verificação, validação e avaliação realizados sobre os requisitos funcionais e atributos de qualidade da solução;
- **Conclusão** - O nono capítulo apresenta as principais conclusões. Explica o trabalho realizado e aponta possíveis funcionalidades a desenvolver futuramente.

Capítulo 2

Análise de Requisitos

Face ao problema proposto, procurou-se entender o seu domínio, demonstrou-se a metodologia a utilizar para criar requisitos funcionais de qualidade, definiram-se um conjunto de requisitos funcionais que se pretendem e apresentaram-se ainda aqueles que se consideram os atributos de qualidade mais importantes para a solução.

2.1 Entender o Domínio do Problema

Para facilitar na tarefa de compreender corretamente o domínio do problema foi escolhida a framework i*[19], proposta em 1994 por Erik Yu e John Mylopoulos [20], pois permite entender melhor as razões por trás do que é pedido, criando uma noção das motivações, intenções, razões e fluxos de entradas-saidas das atividades. A framework possui o *Strategic Dependency Model* e o *Strategic Rationale Model* [21]. A notação utilizada pela framework i* pode ser consultada no apêndice B.

2.1.1 Strategic Dependency Model

O *Strategic Dependency Model* é utilizado para compreender a rede de relações e dependências entre os atores. O modelo construido para o domínio a ser analisado é apresentado na figura 2.1.

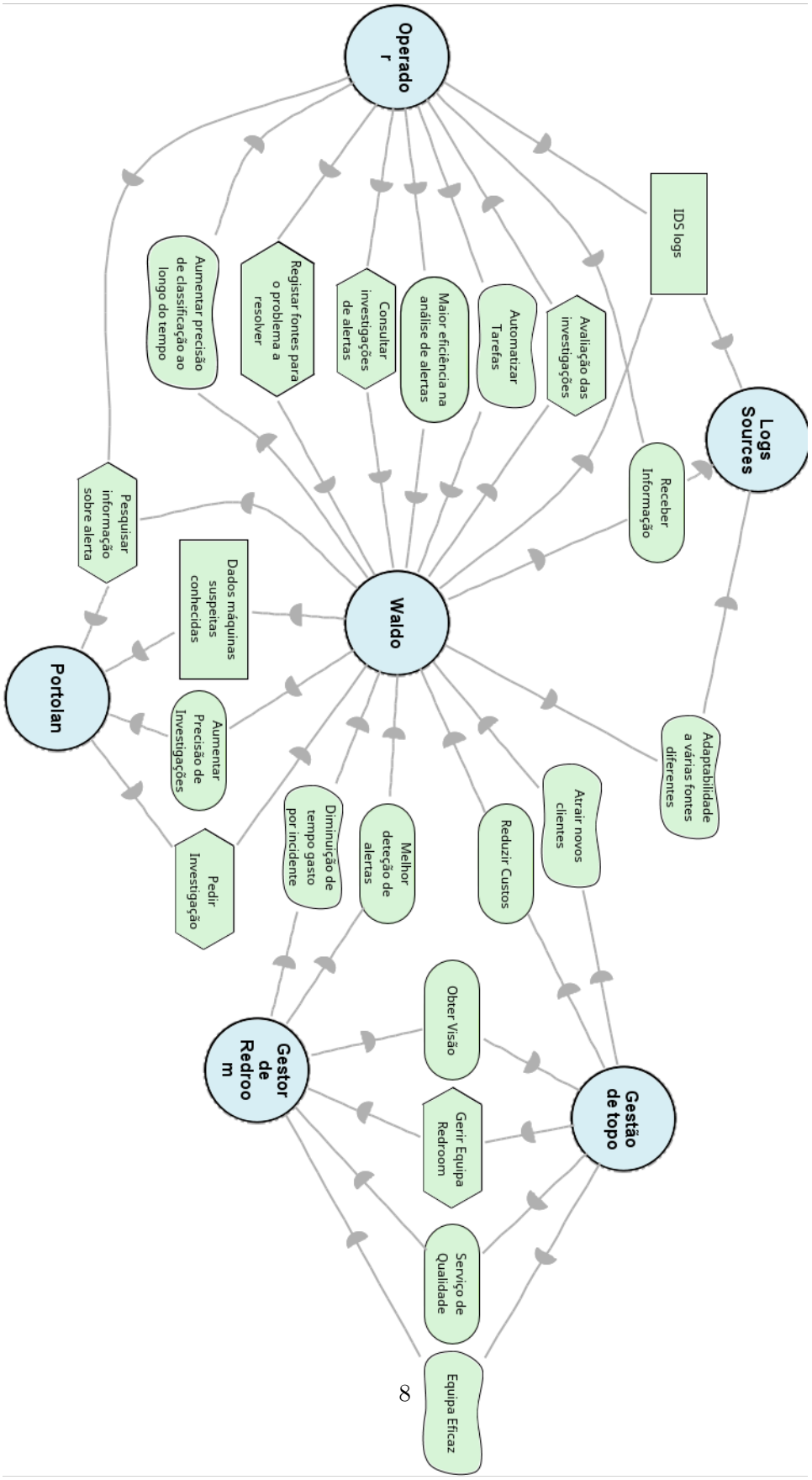


Figura 2.1: Strategic Dependency Model

Como se pode observar existem 6 atores:

- Log Sources;
- Operadores;
- Waldo;
- Portolan;
- Gestão de Topo;
- Gestor de Redroom.

Cada um destes atores possui, tarefas, recursos, interesses e objetivos que dependem uns dos outros. De modo a ajudar na interpretação do diagrama apresentado descrevem-se as dependências do actor principal, o Waldo. Este depende do actor:

- Logs Sources para obter o recurso IDS logs;
- Logs Sources para cumprir o seu objectivo de receber informação;
- Logs Sources para atingir o seu interesse de conseguir adaptabilidade a várias fontes;
- Portolan para conseguir o seu objectivo de aumentar a precisão de investigações;
- Portolan para conseguir o recurso dados de máquinas suspeitas;
- Portolan para executar a tarefa de pesquisar informação sobre alerta;
- Operador para realizar a tarefa avaliação de investigações;
- Operador para atingir o seu interesse de aumentar a precisão de classificação ao longo do tempo.

2.1.2 Strategic Rationale Model

O *Strategic Rationale Model* é aplicado de modo a descrever e apoiar o porquê das relações de cada ator com os outros. Estes relacionamentos são estratégicos no sentido de que cada grupo está concentrado em oportunidades e vulnerabilidades procurando proteger ou alcançar os seus interesses. O modelo construido para o domínio a ser analisado é apresentado na figura 2.2.

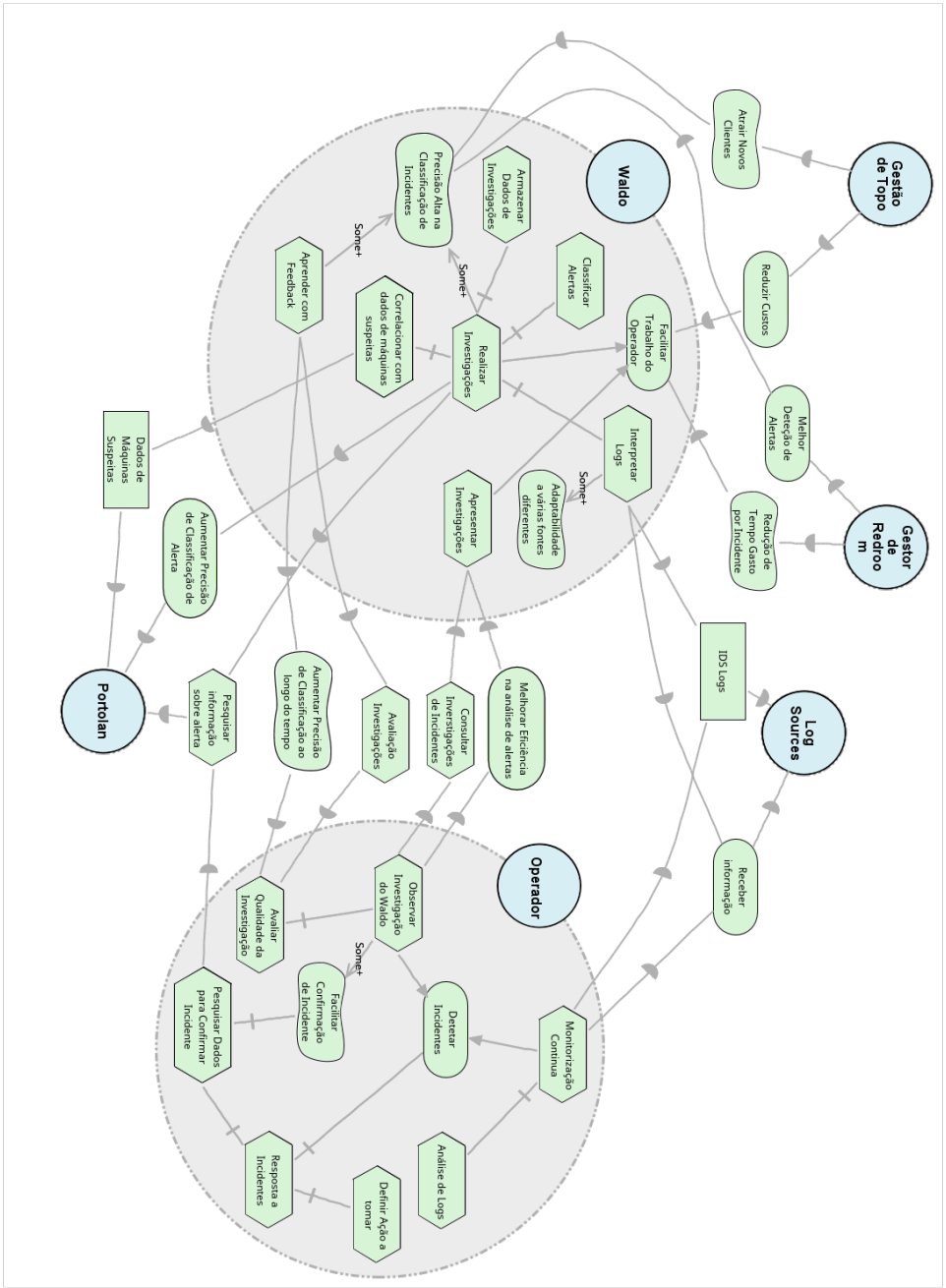


Figura 2.2: Strategic Rationale Model

Para se facilitar na perceção da lógica do modelo é realizada uma interpretação do domínio do ator operador:

- O operador realiza a resposta a incidentes que é composta pelo objetivo

de detetar incidentes, realizando as tarefas de pesquisa de dados, para confirmar a existência do incidente, definindo a ação a tomar;;

- O objetivo do operador de detetar incidentes é obtido pela monitorização continua;
- A tarefa de avaliar a qualidade da investigação é composta pela observação de investigações do Waldo;
- A observação de investigações do Waldo vai ser mais uma fonte para atingir o objetivo de detetar incidentes e vai ainda contribuir para facilitar a confirmação de incidentes;
- Para o operador cumprir a sua tarefa de observar as investigações do Waldo ele depende da tarefa do Waldo de apresentar investigações. Este realiza esta tarefa de modo a atingir o objetivo de melhorar a eficiência na análise de alertas e cumprir a tarefa de consultar investigações de incidentes;
- Para o operador realizar a tarefa da monitorização continua este depende dos logs de IDS recebidos das Log Sources de modo a cumprirem o objetivo de obter informação;
- A tarefa de pesquisa de dados para confirmar incidente vai depender do Portolan para conseguir realizar pesquisas de informação sobre o alerta.

2.2 Escala de Prioridades

Aos requisitos enumerados, foram atribuídas ordens de prioridade, através da seguinte escala:

- **Must Have** - Constitui um requisito fulcral à aplicação quer no âmbito de produto quer no âmbito da dissertação, sem o qual se torna impossível a sua colocação em produção em clientes da empresa.
- **Should Have** - Requisito que embora não seja fulcral ao funcionamento da aplicação, o seu abandono coloca em causa as metas e objetivos para quais a aplicação foi criada, retirando valor ao produto final e a todos os processos de gestão para os quais foi desenhado.
- **Nice to Have** - Requisito de caráter otimizador que embora tenha um valor considerado bastante relevante, não constitui uma prioridade de implementação face aos restantes.

2.3 Estrutura dos Requisitos

De forma a garantir a criação de requisitos bem estruturados, foi utilizada a abordagem EARS[9] para os descrever. Esta abordagem oferece padrões bem definidos para descrever requisitos específicos, apresentando-se aqui as caraterísticas dos mesmos a que estes respondem:

- Ubíquos - Apresentam uma propriedade fundamental do sistema, não precisam de um estímulo para serem ativados e são universais (existem sempre);
- Orientados a Eventos - São iniciados apenas quando é detetado um estímulo;
- Comportamentos Indesejados - Lida com comportamentos indesejados incluindo, defeitos (fault), erros (error), falhas (failure), entre outros;
- Orientados a Estados - Ativado quando o sistema entra num estado específico;
- Características Opcionais - Invocados apenas quando o sistema inclui uma funcionalidade opcional;
- Complexos - Descrevem eventos complexos e condicionais envolvendo a junção de vários estímulos, estados e/ou características opcionais.

Sendo escolhido o padrão a utilizar consoante as características do requisito, é aplicada uma estrutura para a escrita deste requisito. Podem ser consultadas as estruturas de escrita de cada padrão na figura 2.3 retirada de [9].

Pattern Name	Pattern
Ubiquitous	The <system name> shall <system response>
Event-Driven	WHEN <trigger> <optional precondition> the <system name> shall <system response>
Unwanted Behavior	IF <unwanted condition or event>, THEN the <system name> shall <system response>
State-Driven	WHILE <system state>, the <system name> shall <system response>
Optional Feature	WHERE <feature is included>, the <system name> shall <system response>
Complex	(combinations of the above patterns)

Figura 2.3: Estrutura da Escrita dos Padrões EARS

2.4 Requisitos Funcionais

De forma a assegurar que a plataforma cumpre o seu propósito são identificados nesta secção os requisitos funcionais.
Os requisitos que se seguem foram derivados da análise dos diagramas i* apresentados nas figuras 2.1 e 2.2.

- **RF01 - Analisar dados** - Quando receber informações das suas fontes o Waldo deve interpretar e analisar as mesmas;
- **RF02 - Classificar alertas** - Quando recebido um evento e correlacionada a sua informação o Waldo deve classificar esse mesmo alerta como sendo ou não uma intrusão.
- **RF03 - Receber logs** - O Waldo deve estar em espera, sempre pronto a receber logs das suas fontes;
- **RF04 - Apresentar Investigações** - Quando um operador aceder à dashboard o Waldo deve apresentar as investigações realizadas;
- **RF05 - Aprender com feedback** - Quando um operador enviar feedback de uma investigação o Waldo deverá utilizar esse feedback para aprendizagem;
- **RF06 - Aumentar precisão de classificação** - Enquanto o Waldo for utilizado este deve ir aumentando a sua precisão de classificação de alertas;
- **RF07 - Analisar informação do Portolan** - Quando for recolhida informação do Portolan, o Waldo deve conseguir analisar esta informação;
- **RF08 - Pesquisar informação no Portolan** - Quando um alerta for recebido, o Waldo deve pesquisar por informação extra no Portolan;
- **RF09 - Correlacionar dados** - Quando analisada a informação recolhida do Portolan, o Waldo deve ser capaz de correlacionar essa informação com os dados do alerta a investigar;
- **RF10 - Reavaliar alertas** - Quando o Portolan recolher nova informação, o Waldo pode ser notificado de forma a reavaliar os alertas que não sendo uma intrusão antes, agora o podem ser com a nova informação recolhida;
- **RF11 - Apresentar fontes de apoio** - Quando um operador submeter o feedback de uma investigação, incluindo fontes da sua pesquisa, o Waldo numa próxima investigação, se esta for sobre o mesmo tipo de alerta, deve voltar a apresentar essas fontes;

Para completar o sistema pretendido averiguou-se também a necessidade da inclusão dos seguintes requisitos:

- **RF12 - Guardar investigações com alertas suspeitos** - O Waldo deve manter o registo de todas as suas investigações com alertas suspeitos;
- **RF13 - Guardar investigações sem resultados** - Se a investigação não levar a nenhum resultado com informação suspeita o Waldo deve manter esta durante algum tempo para ser utilizada como informação extra para as próximas investigações a realizar.
- **RF14 - Associar pacotes de rede ao alerta** - Quando recebido um alerta por parte de um IDS(Intrusion Detection System) o Waldo deve associar os pacotes de rede desde um pouco antes de ser despoletado o alerta até um pouco depois, de modo a facilitar as investigações.

Para cada um dos requisitos enunciados foi atribuída uma prioridade, como se pode ver na tabela 2.1.

Tabela 2.1: Prioridade de Requisitos

Requisito	Prioridade	EARS
RF03 - Receber logs	Must Have	Ubíquo
RF01 - Analisar dados	Must Have	Orientado a Eventos
RF02 - Classificar alertas	Must Have	Complexo
RF12 - Guardar investigações com alertas suspeitos	Must Have	Ubíquo
RF04 - Apresentar Investigações	Must Have	Orientado a Eventos
RF05 - Aprender com feedback	Should Have	Orientado a Eventos
RF06 - Aumentar precisão de classificação	Should Have	Orientado a Estados
RF07 - Analisar informação do Portolan	Should Have	Orientado a Eventos
RF08 - Pesquisar informação no Portolan	Should Have	Orientado a Eventos
RF09 - Correlacionar dados	Should Have	Orientado a Eventos
RF13 - Guardar investigações sem resultados	Nice to Have	Comportamento Indesejado
RF10 - Reavaliar alertas	Nice to Have	Orientado a Eventos
RF11 - Apresentar fontes de apoio	Nice to Have	Orientado a Eventos
RF14 - Associar pacotes de rede ao alerta	Nice to Have	Orientado a Eventos

2.5 Atributos de Qualidade

De modo a garantir que a aplicação é capaz de cumprir com o seu propósito, ao nível dos seus requisitos não funcionais, descreve-se nesta secção os atributos de qualidade necessários e as metas a atingir em cada um dos mesmos. A partir dos modelos da framework i* apresentados nas figuras 2.1 e 2.2 foram extraídos os atributos:

- Aumentar precisão da classificação ao longo do tempo;
- Precisão alta de classificação de alertas;
- Automatizar tarefas;

- Redução de tempo gasto desde a identificação do alerta até à resolução do incidente;
- Adaptabilidade a várias fontes diferentes.

Há também interesse em assegurar os seguintes atributos de qualidade:

- Modularidade;
- Perfomance.

2.5.1 Aumentar precisão da classificação ao longo do tempo

Quando o Waldo apresenta uma investigação, o operador deve fornecer feedback da qualidade desta. Este feedback vai ser usado para ajustar o modo de análise do sistema e como são ponderadas as informações que são correlacionadas. Este atributo de qualidade está então definido para garantir que o Waldo vai melhorando a uma escala razoável ao longo do tempo, e de modo a serem tomadas medidas caso isso não aconteça. Ao fim de um mês depois do deploy a receber feedback diariamente o Waldo já deve conseguir operar com uma precisão de 10%.

2.5.2 Precisão alta de classificação de alertas

A precisão do Waldo deve ser alta de forma a não perturbar o trabalho do operador com mais informação inútil e para garantir que as investigações criadas apresentam um índice mínimo de alertas e informação correlacionada suspeita. O sistema deve possuir acima de 20% de precisão.

2.5.3 Automatizar tarefas

Para simplificar o processo de pesquisa da informação apresentada, a plataforma deve apresentar pelo menos uma fonte na investigação para ajudar a confirmar os resultados de problemas já conhecidos.

2.5.4 Reduzir tempo gasto desde a identificação do alerta até à resolução do incidente

O aumento da eficiência dos operadores é um dos grandes interesses que levou à implementação do Waldo. Não existe um valor médio, contudo sempre que é detetado um incidente o tempo para o analisar e resolver é bastante inconstante, sendo por vezes muito elevado. Pretende-se que em 80% das vezes que o Waldo faça uma investigação, o tempo para a resolver passe a ser mais baixo do que é atualmente.

2.5.5 Adaptabilidade a várias fontes diferentes

O Waldo não se deve prender a uma só fonte pois pretende-se que este seja incorporado tanto na solução da Dognaedis, como em qualquer outra organização que tenha a mesma necessidade. Deve então conseguir obter informação da elasticstack[22] como também deve conseguir ligar-se a outras fontes com um pequeno número de modificações.

2.5.6 Modularidade

Como já referido pretende-se que o sistema seja integrado na solução da Dogneadis, mas que seja simples de integrar na solução de outras organizações com casos de uso não necessariamente iguais. Desta forma o Waldo tem de ser modular e permitir facilmente adicionar nova lógica e regras para a sua aprendizagem e para a sua solução de problemas.

2.5.7 Perfomance

Dado que a quantidade de incidentes e informação para ser classificada, é bastante elevada em certos curtos espaços de tempo, o Waldo deve conseguir relacionar esta informação em tempo útil. Para conseguir esta análise em tempo útil o Waldo deve conseguir receber um alerta e realizar toda a investigação necessária em média em menos de um segundo.

2.6 Conclusão

Neste capitulo foram aplicadas as técnicas da framework i* para entender melhor o domínio do problema e esta revelou-se extremamente útil para:

- Entender o porquê de cada um dos objetivos pedidos;
- Perceber quais as dependências entre stakeholders e quais os seus interesses;
- Extrair a maioria dos requisitos funcionais e atributos de qualidade.

Foi também utilizada a abordagem EARS que é simples de aprender e permitiu a criação de requisitos funcionais devidamente estruturados, não ambíguos e de fácil interpretação.
Este capítulo permitiu um planeamento mais detalhado do processo de desenvolvimento, fornecendo indicadores valiosos para a definição da arquitetura da plataforma.

Capítulo 3

Estado da Arte

Pretende-se que o Waldo faça a deteção de intrusões e realize investigações utilizando informação de vários componentes da infraestrutura já existente. Os produtos com intuito semelhante serão listados neste capitulo e será realizada a sua comparação com o que é pretendido. Junto com a análise do mercado também serão apresentados artigos que sugerem variadas técnicas interessantes para a implementação do sistema.

3.1 Produtos

- **Hexadite[23]** - É uma solução de *Security Orchestration* que possui a linha completa de receber informação de várias fontes, correlacionar essa informação de modo inteligente, detetar intrusões, aplicar controlos e respostas para cada um dos problemas identificados automaticamente, prover informação ao operador do que está a ser feito e gerar tickets de modo automático.
  Este sistema possui como pontos mais apelativos a capacidade de receber informação das maiores plataformas de IDS disponiveis no mercado, conseguir correlacionar a sua informação de forma eficaz e gerar uma resposta automática para os problemas identificados.

- **Splunk[24]** - Sistema capaz de deteção e triagem de intrusões, monitorização em tempo real, apresenta investigações com o rastreamento das atividades relacionadas ao incidente e possibilita a utilização de várias fontes de inteligência para realizar decisões com a melhor informação possível.
  Possui como pontos mais interessantes, a utilização da modelação de comportamentos através de algoritmos de aprendizado automático, ganhando a capacidade de se adaptar, com aprendizagem não supervisionada, em vez da utilização das abordagens comuns, como alertas gerados a partir de regras ou de assinaturas, ou a demonstração de investigações ao operador com os dados relacionados ao incidente.

- **FireEye[25]** - Também um sistema de *Security Orchestration*, apresenta investigações de intrusões, faz correção e proteção de dispositivos ligados à rede que estejam infetados, realiza análise de software malicioso em ambiente seguro, possui fontes de dados próprias e capacidade de análise dessas fontes. Tem como pontos mais apelativos para o Waldo a junção de inteligência de várias fontes para permitir uma mais fácil análise das investigações por parte do operador .

De modo a facilitar a visualização das diferenças, vantagens e desvantagens de cada um dos produtos, para os requisitos pretendidos, é apresentada uma comparação dos mesmos na tabela 3.1.

Nome	Hexadite	Splunk	FireEye	Waldo
Analisar dados	√	√	√	√
Classificar alertas	√	√	√	√
Receber logs	√	√	×	√
Apresentar Investigações	√	√	√	√
Aprender com feedback	×	√	×	√
Aumentar precisão de classificação	×	√	×	√
Analisar informação do Portolan	×	×	×	×
Pesquisar informação no Portolan	×	×	×	×
Correlacionar dados	×	×	×	×
Reavaliação de alertas	×	×	×	×
Apresentar fontes de apoio	√	√	√	√
Guardar investigações com alertas suspeitos	√	√	√	√
Guardar investigações sem resultados	-	-	-	×
Associar pacotes de rede ao alerta	-	-	-	×

Tabela 3.1: Comparação Produtos
"√" - Possui funcionalidade
"×" - Não possui funcionalidade
" - " - Não se consegue verificar a existência da funcionalidade

3.2 Artigos

Para entender as técnicas mais atuais utilizadas neste tipo de sistema, são sumariados nesta secção artigos de sistemas de deteção de intrusões que utilizam aprendizado automático.

3.2.1 Outside the Closed World: On Using Machine Learning For Network Intrusion Detection [1]

O artigo fala sobre a premissa de que quase todos os sistemas de deteção de intrusões no mercado, são baseados em *misuse-detection*, ou seja, baseiam-se em descrições precisas de comportamentos de malware conhecido, quando técnicas de aprendizado automático poderiam ser aplicadas e garantir uma diferente capacidade de deteção que será benéfica para este tipo de sistema. É falada da *closed world assumption*, este termo significa que este tipo de sistema pode ser treinado apenas com dados normais para entender qual o padrão normal e tudo o que for fora desse padrão são dados perigosos. O artigo explica que assumir esta lógica pode ser errado pois um comportamento fora do normal na rede não é assim tão incomum e não significa necessariamente uma intrusão, o que pode levar à geração de falsos positivos. É apresentado o risco elevado dos sistemas de deteção de intrusões com aprendizado automático pois estes não devem permitir a existência de muitos falsos positivos nem falsos negativos. É reforçado que não há datasets públicos de confiança para serem utilizados para uma boa previsão e é colocado um grande ênfase na análise tanto de falsos positivos, como verdadeiros positivos e verdadeiros negativos.
Para exemplificar a necessidade desta análise foi apresentada no site [26] a história:
Em 1980 o Pentágono criou um projeto em que foi treinada uma rede neuronal para detetar tanques em fotos. Numa avaliação inicial esta foi capaz de separar corretamente fotos com tanques das fotos em que não havia tanques. No entanto, o dataset utilizado continha uma característica subtil, as fotos que continham tanques foram tiradas num dia em que estavam nuvens e céu mais cinzento, enquanto as fotos sem tanques tinham sido tiradas num dia com o céu azul. Quando chegou o dia de apresentar a ferramenta foram experimentadas novas imagens de tanques e os resultados obtidos eram aleatórios. Acontece que o que a rede neuronal tinha aprendido a distinguir era se estava ou não sol.

"The military was now the proud owner of a multi-million dollar mainframe computer that could tell you if it was sunny or not." - Neil Fraser[26]

3.2.2 BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation [2]

Este artigo apresenta uma abordagem ao mesmo nível que o Waldo se vai posicionar. O sistema abordado no artigo, recebe os alertas gerados por IDS já existentes e tenta correlacionar a informação. O objetivo é conseguir entender se existe alguma infeção por um bot. Para tal, é necessário entender o ciclo de vida de uma infeção normal de um bot, de forma a conseguir

compreender bem o processo. Este ciclo de vida foi inicialmente dividido em 5 passos:

- **Passo 1** - Procura um alcance de portos de serviços de rede para tentar encontrar algum que esteja exposto e vulnerável, ou procura respostas de *trojans backdoors*, que possam ser utilizados para entrar na máquina infetada. Se o bot receber uma resposta de ligação, ativa um programa que explore os serviços vulneráveis, ou entra no host comunicando com o *trojan backdoor* instalado;

- **Passo 2** - Já infetada, a vitima executa um *shell script* para abrir um canal de comunicação com a máquina do atacante de forma a descarregar o código binário do bot;

- **Passo 3** - O bot insere-se no processo de arranque do sistema, desliga o software de segurança, realiza um scan à rede local à procura de NetBIOS shares e procura outros malwares para garantir que é o único instalado na máquina;

- **Passo 4** - De seguida a vitima passa a ser um bot, pois é estabelecida uma ligação a um servidor de controlo de botnet, a partir de algum tipo de protocolo (como IRC por exemplo);

- **Passo 5** - Finalmente o novo bot infectado coloca um porto em escuta a aceitar updates binários e começa a realizar scans na rede, de modo a encontrar novas vitimas para a botnet.

Neste passos foram observadas na vitima as cinco seguintes transações:

- **Transação 1** - Comunicação da rede externa para a máquina realizando um scan aos serviços disponiveis;

- **Transação 2** - Comunicação da rede externa para a máquina explorando um serviço vulnerável;

- **Transação 3** - Comunicação da máquina para a rede externa realizando um download de um payload binário;

- **Transação 4** - Comunicação da máquina para a rede externa falando com uma rede de C&C(comando e controlo) de bots(botnet);

- **Transação 5** - Comunicação da máquina para a rede externa procurando novas vitimas para infectar.

São utilizadas máquinas Windows para testar. Foi utilizado o Snort[27] com regras personalizadas para deteção de malware e foram criados dois plugins. O primeiro plugin (SLADE) realiza análise de payload do tráfego que entra na rede, procurando diferenças na distribuição de bytes em relação

aos protocolos selecionados. O segundo plugin (SCADE) realiza scans complementares a portos, para tráfego que entra e sai da rede. O BotHunter associa tráfego que entra na rede e alarmes de intrusões com padrões de comunicação indicativos de intrusões de malware de tráfego que sai da rede, obtendo um total de 95,1% de verdadeiros positivos em 2019 infeções por bots.

3.2.3 Alert Correlation Algorithms: A Survey and Taxonomy [3]

O artigo começa por explicar que os IDS geram alertas por prioridade, filtrando os eventos de rede que conseguirem. O volume de alertas apresentado mesmo após filtrado é alto o que inviabiliza a análise completa a todos os alertas gerados, como tal podem ser perdidos dados importantes.
De seguida apresenta várias classes de algoritmos para correlacionar alertas:

- **Baseado em semelhanças** - Tem como objetivo o clustering de alertas o que lhes garante a vantagem de conseguirem agrupar alertas sem tipos de ataques definidos;
- **Baseado em conhecimento** - Os algoritmos baseados em conhecimento oferecem um grande capacidade de deteção, contudo não conseguem responder a ataques não conhecidos e envolvem uma manutenção do conhecimento que lhes é passado. Estes dividem-se em dois grupos:
  - **Pré-requisito e Consequência** - Esta abordagem é de mais alto nível, associa cada incidente a outros incidentes por uma rede de conjunções e disjunções gerando uma rede de possíveis ataques. Neste tipo de algoritmos não é necessária a definição de cada cenário de ataque, contudo é necessário a definição de pré-requisitos que podem gerar os ataques e possíveis resultados desses ataques;
  - **Cenário** - São baseados na ideia de que as intrusões são constituídas de vários passos que devem ocorrer para o ataque obter sucesso, sendo como tal necessário ir criando e mantendo os cenários necessários para cada tipo de ataque.
- **Baseado em estatística** - Vão atrás do conceito de que ataques relevantes vão ter dados estatísticos semelhantes e como tal podem ser associados. Guardam relações de causas dos vários incidentes e analisam as suas frequências no sistema, usando então a análise estatística para gerar as etapas de um ataque. Não necessitam de conhecimento de ataques para operar, contudo não funcionam em vários domínios e apresentam uma taxa de erro maior.

Na figura 3.1, verifica-se a comparação entre as caracteristicas de cada abordagem. Outra comparação consoante vários factores desde a precisão, flexibilidade, extensibilidade, memória necessária, capacidade de computação, capacidade de paralelização até ao conhecimento necessário entre os vários tipos de algoritmos, pode ser analisada na figura 3.2. Ambas as figuras foram retiradas do artigo.

Characteristic	Similarity-based	Knowledge-based	Statistical-based
Combining alerts from various sensors	Yes	Yes	No
Requiring Prior knowledge	Yes	Yes	No
Detecting false alerts	Yes	Yes	Guessing
Detecting multi-stage attacks	Hardly	Yes	Guessing
Find new attacks	Yes	No	Yes
Error rate	Average	Low	High

Figura 3.1: Comparação de características de correlação de alertas

		Accuracy	Flexibility	Extendibility	Required Memory	Computation Power	Parallelizing	Knowledge base
Similarity-based	Simple Rules	A	H	H	A	A	H	-
	Hierarchical Rules	A	H	H	A	A	H	AD
	Machine Learning (Decision Tree)	A	A	A	A	A	A	-
	Machine Learning (Re-creation)	A	A	A	L	H	H	-
	Machine Learning (Verification)	A	A	A	H	H	L	-
KB	Prerequisites/Consequences	H	H	H	H	L	L	AM
	Scenario	H	H	H	A	A	L	H
Statistical-based	Statistical Traffic Estimation	A	H	A	L	H	H	-
	Statistical Traffic Estimation (Association Rules)	A	L	L	A	A	A	-
	Causal Relationship Estimation (Ganger Test)	A	L	L	A	A	H	-
	Causal Relationship Estimation (Markov Model)	A	L	L	A	A	A	-
	Reliability Degree Combination	A	L	L	A	A	H	-

H	High
A	Average
L	Low
AD	Require Attack Definition
AM	Require Alert Meaning

Figura 3.2: Comparação de algoritmos de correlação de alertas

3.2.4 Using unsupervised learning from Network Alert Correlation [4]

O artigo aborda uma técnica para correlacionar alertas de IDS de rede, utilizando machine learning com aprendizagem não supervisionada. Esta técnica baseia-se na idea de que os ataques podem ser decompostos em várias etapas, que correspondem a uma ação no grande plano dos atacantes e que estas etapas consistem de um grande numero de alertas de IDS. A forma de atuar divide-se então em dois passos:

- É realizado o clustering dos alertas para identificar caracteristicas que os diferenciem. Para este clustering são utilizados os protocolos de

pacotes IP, criando clusters independentes do IDS que os gerou;

- De seguida procura-se entender as tendências dentro de cada um dos grupos de alertas já relacionados, realizando clustering agora já usando os dados de cada um dos alertas, de modo a corresponder a uma etapa do ataque e também de forma a que etapas semelhantes tenham dados próximos.

Foram apresentados três algoritmos e comparados os seus resultados aplicando uma combinação para treino e teste de datasets do DARPA e do incidents.org:

- **AA** - Utilizando a técnica *Autoassociator*, descrita em [28], obteve-se o melhor resultado com uma taxa de erro de um pouco mais baixa que 20%;
- **EM** - O *Expectation-Maximization* ficou em segundo lugar com uma taxa de erro de 30%;
- **SOM** - A abordagem *Self-organizing Maps* ficou um último com uma taxa de erro de 40%;

É interessante notar que o dataset continha 13 tipos de ataque e o melhor algoritmo criou 21 clusters que representam diferentes tipos de ataque. Isto significa que o sistema pode ter criado erros de separação, ou seja, separado alertas que tinham a mesma classificação em termos do tipo de ataque. 54 de 500 Alertas estavam noutro cluster em relação ao que deveriam estar, contudo os elementos desse grupo estavam em clusters com alertas que lhes estavam associados. Apenas 4 elementos em 500 estavam em clusters não relacionados o que é bastante positivo pois erros de separação são muitos menos graves que erros de clustering.

3.2.5 An Online Adaptive Approach to Alert Correlation [5]

O artigo apresenta uma abordagem baseada em análise estatística utilizando redes bayesianas. Foram implementados dois módulos, o módulo offline responsável por correlação e seleção das features e o módulo online que é utilizado para aplicar a correlação de alertas e extrair novas estratégias de ataque *on-the-fly*.
O funcionamento desta abordagem baseia-se em 3 passos principais:

- **Análise do comportamento dos alertas** - Uma tabela de correlação é mantida para monitorizar mudanças repentinas no comportamento dos alertas. Esta probabilidade é de seguida comparada com a apresentada na tabela de relevância, se a probabilidade de ocorrência de alertas não corresponder à apresentada na tabela de correlacionamento então esta informação é colocada na tabela de correlação temporária;

- **Fusão de alertas** - Antes de a probabilidade de correlação de hiper alertas(conjunto de alertas) poder ser calculada, são emparelhados os alertas que anteriormente mostraram uma forte ligação tanto na tabela de correlação como na tabela de correlação temporária. À primeira vista pode não existir correlação entre alertas, então é aplicada uma probabilidade como limite que permite a navegação para pares de alertas com fortes relacionamentos e que são mais prováveis de representar um passo de ataque;

- **Construção do cenário de ataque** - Os cenários de ataque são gerados baseados nos pares de causas de alertas relacionados.

Finalmente foram apresentados dois testes, um offline e outro online. O offline foi realizado com 1508 alertas do Snort, gerando 729 pares de alertas destes apresentou 226 pares de alertas com causas relacionadas, sendo destes 191 pares corretamente correlacionados e 36 incorretamente correlacionados. Existiam 196 pares correlacionados no total, então obteve-se uma taxa de 96,5% de verdadeiros positivos positivos e uma taxa de 15,9% de falsos positivos. De seguida foi utilizado o método online para verificar o comportamento em tempo real. Foi analisada uma hora de alertas de Snort em foram gerados 221 novos alertas, destes foram encontrados 2 novos tipos de ataque. Foi realizado o recalculo da probabilidade de correlacionamento entre estes dois novos tipos e todos os já existentes, aumentando a taxa de verdadeiros positivos para 96,1% e diminuindo a de falsos positivos para 14%.

3.2.6 Application of Case Based Reasoning in IT Security Incident Response [6]

O artigo explora a possibilidade de utilização de utilização de um sistema CBR para ajudar ao tratamento e resposta de incidentes. Este apresenta o caso especifico da Malaysia CERT[29] que no ano de 2014 detetou 11918 incidentes, um aumento de 12% em relação a 2013. Com este volume de incidentes, e sendo este crescente de ano para ano, o tempo de resposta a ataques e incidentes tem de ser levado seriamente de modo a garantir a segurança das organizações.
Os maiores problemas encontrados foram:

- Os peritos em reposta e tratamento de incidentes passam a maioria do seu tempo em tratamento e resposta a incidentes. Devido ao grande volume destes torna-se difícil conseguir responder a todos em tempo útil, tornando-se essencial a partilha de técnicas e conhecimento;

- O comportamento dinâmico das ameaças de segurança, visto que é sabido que os atacantes partilham bastante informação entre eles sobre programas para explorar vulnerabilidades e *0-days*. Existindo até tipos

de mercados negros onde são vendidos dados sensíveis, conjuntos de software para explorar vulnerabilidades e templates para *malware*. Os operadores responsáveis por detetar e tratar os incidentes regularmente têm outros papéis nas organizações, possuindo um tempo limitado para conseguir detetar e tratar os incidentes.

Deste modo considera-se bastante importante conseguir:

- Diminuir a probabilidade de os operadores cometerem erros;
- Oferecer uma linha de referência enquanto se responde a incidentes de modo a conseguir uma resposta mais rápida e completa;
- Facilitar a obtenção de conhecimento quando o próximo operador se deparar com o mesmo problema ou um problema semelhante;
- Que os novos operadores de segurança consigam o conhecimento e técnicas básicas necessárias ao tratamento de incidentes o mais rapidamente possível.

Para isto foi escolhido um sistema CBR pois este é apropriado para ser aplicado em domínios:

- Em que os casos passados estejam disponíveis;
- Que não sejam bem compreendidos;
- Não estruturados;
- Mal definidos.

Este sistema baseia-se nas ocasiões positivas passadas, representadas na forma de casos. Um caso é um pedaço de informação que descreve uma experiência passada em que ocorreu um problema e qual a respetiva solução adotada. Para descrever o problema e a solução de cada caso foram utilizados os atributos:

- ID Caso;
- Problema:
  - Tempo do incidente;
  - Categoria do incidente;
  - Subcategoria do incidente;
  - URL reportado;
  - IP reportado.
- Solução:

- Tempo de resposta;
- Tipo de ação;
- Passos de erradicação;
- Contactos notificados;
- Entidades notificadas;
- Recomendações.

Os casos são guardados e indexados, sendo bastante importante a escolha dos atributos que representem estes. Esta escolha é importante pois vai permitir ao operador encontrar no sistema CBR o conhecimento mais apropriado para a resolução do caso atual.
O artigo conclui indicando que com casos suficientes o sistema seria capaz de responder de forma autónoma a um grande numero de incidentes em situações de crise e referindo que se pretende como futuro trabalho a implementação do sistema.

3.3 Conclusão

A partir dos produtos analisados pode-se entender que já existem algumas soluções para o que é pretendido, contudo são soluções que replicam partes já existentes na empresa, não permitem uma fácil integração com a plataforma de ciber inteligência da organização que é uma grande mais valia no correlacionamento de alertas e limitam/não possuem a capacidade de aprendizagem ao longo do tempo. Desta forma foram tidos em conta os seguintes pontos das mesmas de forma a conseguir uma melhor análise do possível futuro do Waldo:

- Capacidade de receber informação das maiores plataformas de IDS disponiveis no mercado;
- Modelação de comportamentos através de algoritmos de aprendizado automático;
- Junção de inteligência de várias fontes.

Visto que o correlacionamento de alertas é o maior risco que pode levar o projeto a falhar foram conseguidas, a partir da leitura e análise de vários artigos, técnicas muito interessantes, cada uma com as suas vantagens e desvantagens, que vão ser aplicadas e combinadas no módulo de correlacionamento de alertas do Waldo.
Este capitulo ajudou bastante a entender o que já existe no mercado, a aumentar o conhecimento do domínio e a perceber o funcionamento das técnicas de correlacionamento e classificação de alertas.

Capítulo 4

Planeamento

Neste capítulo descrevem-se as principais fases do projeto, são abordados os maiores riscos que podem ser encontrados e procuradas soluções para a sua mitigação, é apresentado o planeamento temporal das tarefas a realizar inicialmente proposto pela organização e de seguida o planeamento mais detalhado proposto pelo autor.

4.1 Fases do Projeto

Como destacado na proposta de estágio apresentada no apêndice D, o estágio consistirá nas seguintes atividades e suas respetivas tarefas:

- **T1** – Estudo de Plataformas da Empresa – Nesta fase deve-se perceber a forma de operação das plataformas de monitorização de rede e a informação que estas fornecem ao operador de monitorização para a realização do seu trabalho.
- **T2** – Levantamento de Requisitos – Dadas as plataformas e registos existentes, deverá perceber quais os requisitos necessários ao registo de informação e comportamento, correlação de eventos e apresentação de informação ao operador, bem como receção e tratamento de feedback. Os requisitos necessários devem ser procurados em ferrementas já existentes no mercado e deve ainda neste passo ser definida a arquitetura a implementar.
    - **2.1** - Entender o domínio do problema;
    - **2.2** - Definir requisitos funcionais e não funcionais;
    - **2.3** - Planeamento do projeto;
    - **2.4** - Definição da arquitetura.
- **T3** – Desenvolvimento – Desenvolvimento dos requisitos identificados em T2.

T3.1 - Implementar RF3 Receber logs;

T3.2 - Implementar RF1 Analisar dados;

T3.3 - Implementar RF2 Classificar alertas;

T3.4 - Implementar RF12 Guardar investigações com alertas suspeitos;

T3.5 - Implementar RF4 Apresentar Investigações;

T3.6 - Implementar RF5 Aprender com feedback;

T3.7 - Implementar RF6 aumentar precisão de classificação;

T3.8 - Implementar RF7 Analisar informação do Portolan;

T3.9 - Implementar RF8 Pesquisar informação no Portolan;

T3.10 - Implementar RF9 Correlacionar dados;

- **T4** – Testes – Testes funcionais e atributos de qualidade ao trabalho desenvolvido ao longo das tarefas anteriores.

  T4.1 - Testes de aceitação aos requisitos funcionais;

  T4.2 - Testes de validação de Aumentar precisão da classificação ao longo do tempo;

  T4.3 - Testes de validação de Precisão alta de classificação de alertas;

  T4.4 - Testes de validação de Automatizar tarefas;

  T4.5 - Testes de validação de Reduzir tempo gasto desde a identificação do alerta até à resolução do incidente;

  T4.6 - Testes de validação de Adaptabilidade a várias fontes diferentes;

  T4.7 - Testes de validação de Modularidade;

  T4.8 - Testes de validação de Perfomance.

- **T5** – Relatório – Relatório de Estágio.

4.2 Riscos

Ao observar o plano, iniciou-se uma procura pelo que pode falhar neste projeto. Desta análise surgiram os riscos que se seguem:

- **Não conseguir criar ligações e relações entre a informação recolhida** - Podem não ser encontradas relações entre os dados o que leva a ter de ser realizada várias vezes a fase de análise e tratamento de dados;

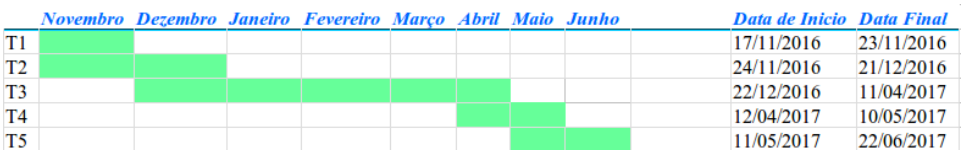
- **Capacidade de classificação do Waldo não atingir níveis aceitáveis** - Se o Waldo não for capaz de classificar incidentes corretamente, este não vai ter utilidade para a equipa;
- **Necessidade de uma grande quantidade de dados** - Pode existir a falta de dados para treinar o Waldo, pois de forma a conseguir um treino de qualidade seria mais fiável a realização do treino com um maior volume de dados de modo a aumentar a capacidade de generalização do sistema;
- **Waldo enviar demasiados falsos positivos** -Quando o Waldo apresentar investigações, estas podem não ser intrusões, se esta situação se repetir muitas vezes o operador começa a dar menos atenção ao Waldo.
- **Atraso no desenvolvimento** - Dado que é necessário correlacionar variada informação, juntar dados de várias fontes, realizar um pré-processamento dos dados e é necessário treinar o Waldo até ser atingida uma boa precisão, o desenvolvimento pode sair do tempo planeado ou não apresentar resultados positivos, pois este é um processo iterativo que consoante a relevância dos dados pode ter de repetido várias vezes, tornando-se bastante demorado.

De modo a tentar aumentar o sucesso do projeto foram definidas as seguintes medidas para a mitigação dos riscos apresentados:

- **Não conseguir criar ligações e relações entre a informação recolhida** - Quando for recolhida informação das várias fontes esta deve passar por várias formas de pré-processamento de forma a procurar correlações entre os dados e conseguir que esta seja melhor utilizada pelo Waldo;
- **Necessidade de uma grande quantidade de dados** - Caso haja uma baixa quantidade de dados pode ser utilizada a abordagem de cross-validation para aumentar o volume de dados, contudo esta abordagem deve ser utilizada apenas se necessário pois aumenta a probabilidade de ocorrer overfitting como demonstrado em [30];
- **Waldo enviar demasiados falsos positivos** - De modo a reduzir o numero de falsos positivos deve ser tido como objetivo do treino precisão em vez de recall;
- **Atraso no desenvolvimento** - A parte mais perigosa do projeto deve ser a primeira a ser abordada de forma a que caso hajam deslizes ou maus resultados estes se reflitam na perca de funcionalidades menos importantes e não no core necessário ao funcionamento básico do Waldo.

4.3 Planeamento

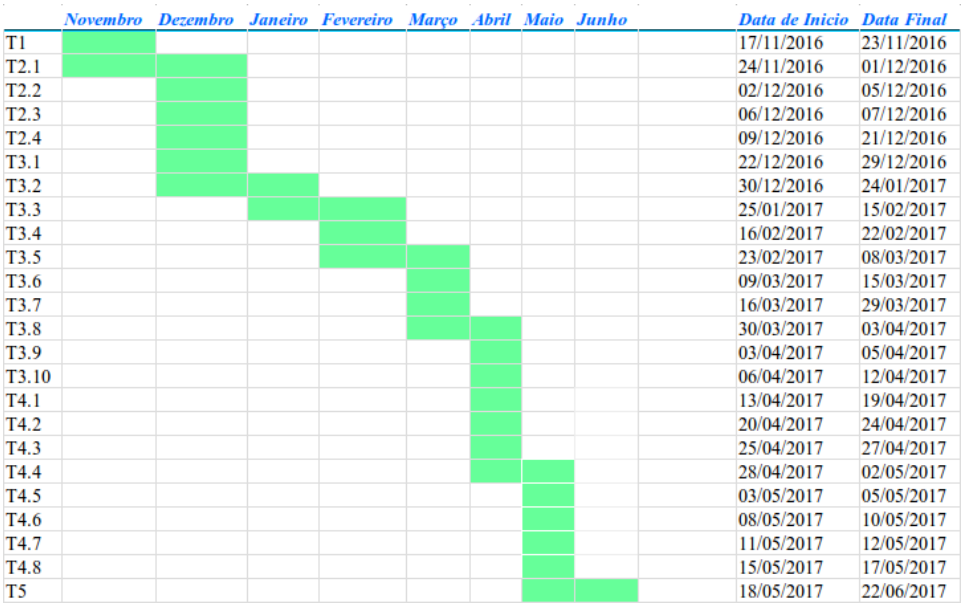
Tendo em conta que o estágio possui uma duração de 1200, começou na data 17/11/2016 e termina a 22/06/2017, é demonstrado na figura 4.1 o espaço temporal definido para cada tarefa que foi apresentada na proposta de estágio.



	Novembro	Dezembro	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Data de Inicio	Data Final
T1									17/11/2016	23/11/2016
T2									24/11/2016	21/12/2016
T3									22/12/2016	11/04/2017
T4									12/04/2017	10/05/2017
T5									11/05/2017	22/06/2017

Figura 4.1: Planeamento Inicial do Projeto

De forma a estabelecer uma timeline mais precisa de objetivos e de material a entregar, em cada uma das etapas, foi ajustado o planeamento para se enquadrar com os requisitos identificados na fase de análise de requisitos, como se poder ver na figura 4.2.



	Novembro	Dezembro	Janeiro	Fevereiro	Março	Abril	Maio	Junho	Data de Inicio	Data Final
T1									17/11/2016	23/11/2016
T2.1									24/11/2016	01/12/2016
T2.2									02/12/2016	05/12/2016
T2.3									06/12/2016	07/12/2016
T2.4									09/12/2016	21/12/2016
T3.1									22/12/2016	29/12/2016
T3.2									30/12/2016	24/01/2017
T3.3									25/01/2017	15/02/2017
T3.4									16/02/2017	22/02/2017
T3.5									23/02/2017	08/03/2017
T3.6									09/03/2017	15/03/2017
T3.7									16/03/2017	29/03/2017
T3.8									30/03/2017	03/04/2017
T3.9									03/04/2017	05/04/2017
T3.10									06/04/2017	12/04/2017
T4.1									13/04/2017	19/04/2017
T4.2									20/04/2017	24/04/2017
T4.3									25/04/2017	27/04/2017
T4.4									28/04/2017	02/05/2017
T4.5									03/05/2017	05/05/2017
T4.6									08/05/2017	10/05/2017
T4.7									11/05/2017	12/05/2017
T4.8									15/05/2017	17/05/2017
T5									18/05/2017	22/06/2017

Figura 4.2: Planeamento do Projeto

4.4 Conclusão

Devido ao conhecimento do domínio do problema obtido na análise de requisitos no capitulo 2 foi possível perceber os riscos associados à implementação do sistema e conhecer melhor as tarefas de modo a criar um planeamento mais detalhado e preciso. Verifica-se que o planeamento não engloba todos os requisitos listados no capitulo 2 de modo a permitir a implementação de cada um dos requisitos de forma cuidada e garantindo a sua qualidade, podendo estes ser implementados caso o projeto termine antes do planeado.

Capítulo 5

Arquitetura do Sistema

Neste capitulo é apresentada a proposta arquitetural para o sistema, de acordo com os requisitos especificados no capitulo 2.

5.1 Enquadramento da Solução na Infraestrutura de Alertas

De modo a entender onde se enquadra o sistema a desenvolver é demonstrada a representação da infraestrutura de registo e report de alertas na figura 5.1 e de seguida é realizada uma breve descrição de cada um dos módulos apresentados.

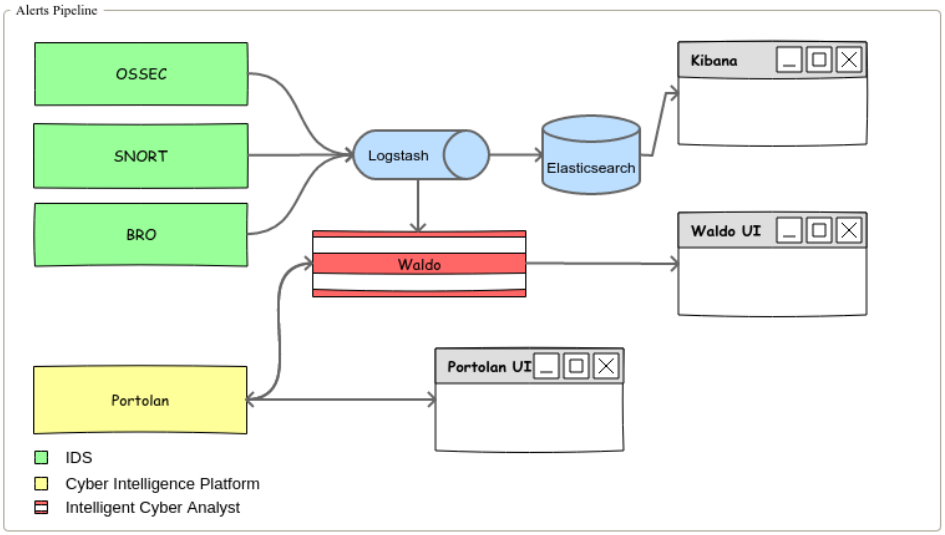


Figura 5.1: Enquadramento do Sistema

Descrição de cada um dos módulos:

- OSSEC[31] - Esta ferramenta dedica-se à monitorização de logs, processos e *rootcheck*, lançando um log de alerta e enviando um email à equipa de IT quando é detetado um ataque ou comportamento suspeito. É cross-platform tendo como tal análises nos diferentes ficheiros de logs dos variados sistemas;

- SNORT[27] - É um software que faz análise de tráfego da rede em tempo real realizando análise de protocolos, procura de conteúdos e compara este tráfego com regras adaptadas para os tipos de ataques já conhecidos. É o sistema de prevenção de intrusões mais utilizado a nível mundial[32];

- BRO[33] - Define-se como uma framework de análise da rede. Possui analisadores para os vários protocolos e uma linguagem própria para a criação de plug-ins, o que permite a sua adaptabilidade para resolver todo o tipo de problemas. Possui interfaces documentadas para integração com outras aplicações possibilitando a troca de informação em tempo real;

- A Elastic Stack[22] oferece a junção de dados estruturados e não estruturados de qualquer fonte, cria uma fácil pesquisa e análise dos dados utilizando a seguinte combinação de ferramentas:

  - Logstash[34] - É uma pipeline que consegue receber dados de uma grande quantidade de fontes simultaneamente, permitindo transformar e centralizar os mesmos numa base de dados à escolha;
  - Elasticsearch[35] - Normalmente é o local utilizado pelo Logstash para centralizar os dados pois armazena os dados e possui uma grande capacidade de pesquisa e análise dos dados através das suas queries especificas[36].
  - Kibana[37] - Permite a visualização dos dados armazenados no Elasticsearch com variados tipos de filtros apresentando grande quantidade e qualidade nos componentes gráficos para representar e interpretar os dados.

- Portolan - Foi desenvolvido na Dognædis, sendo uma plataforma de segurança totalmente integrável e independente de fontes, possui diferentes mecanismos de correlação para apoiar a mitigação de incidentes de modo pró-ativo. Vai comunicar com o Waldo de forma a se conseguir uma análise mais avançada e fiável quando detetado algum evento suspeito.

5.2 Desenho e Especificação da Arquitetura Externa

De forma a entender estrutura do sistema de forma geral, nesta secção é apresentada a arquitectura de alto nivel do Waldo como se pode ver na figura 5.2.

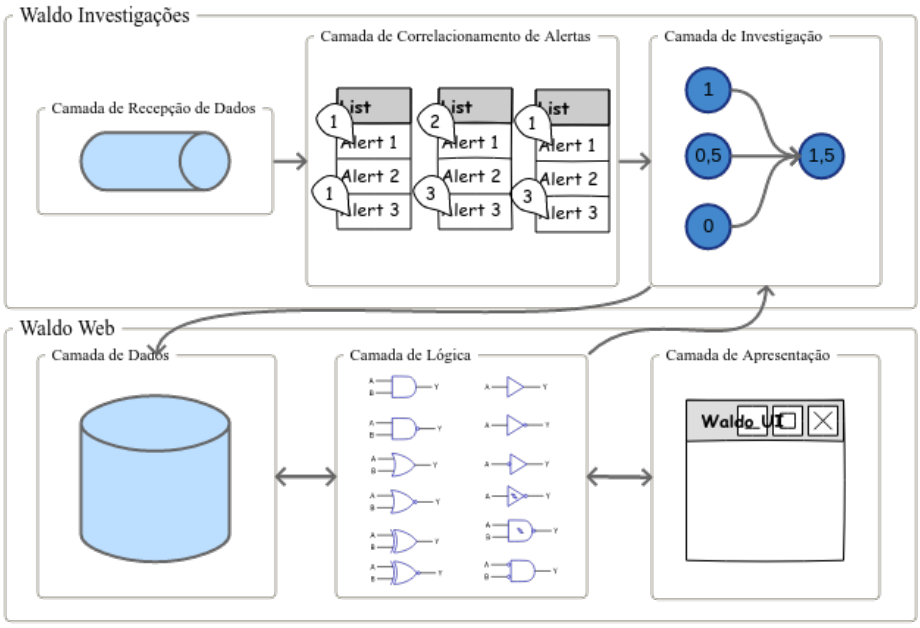


Figura 5.2: Arquitectura Externa do Sistema

O Waldo separa-se em dois principais componentes:

- O de investigações que se especializa na realização de investigações através da correlação e classificação de alertas;
- O web apresenta as investigações ao operador, facilita a sua pesquisa pelos dados de cada investigação e recolhe feedback deste.

De modo a garantir conciliação com vários dos atributos de qualidade apresentados em 2.5 procurou-se a escolha de modelos arquiteturais standard pois estes já foram bastante testados e oferecem estruturas bem organizadas e facilmente reconhecíveis.
Após leitura e análise de [38], escolheu-se uma arquitetura do tipo Mediador, pois para as investigações são necessários vários passos para processar os alertas. Pretende-se com esta escolha, tirar vantagem da paralelização do correlacionamento dos alertas e dado que os correlacionadores possuem um estado em mudança continua é também importante a gestão da ordem

das etapas espoletadas. A gestão dos passos do mediador será então realizada por uma máquina de estados que irá controlar o fluxo de alertas e investigações. A estrutura do componente de investigações do Waldo não depende apenas do Mediador, então este foi dividido em três camadas:

- **Camada de Receção de Dados** - Responsável por receber, gerir e validar os alertas recebidos pelas variadas fontes;
- **Camada de Correlacionamento de Alertas** - Esta camada aplica várias técnicas de correlacionamento de alertas recebidos da camada de receção de dados, procurando encontrar padrões e relações entre os variados alertas. Estas técnicas utilizadas realizam uma análise continua e podem evoluir ao longo do tempo de análise;
- **Camada de Investigação** - A partir dos dados do correlacionamento de alertas esta camada realiza o ponderamento final dos resultados obtidos e estrutura a investigação para ser armazenada para consulta posterior. A metodologia de ponderamento pode variar a partir do feedback passado pelo operador;

O Mediador controla a busca de dados da camada de receção de dados, possui todo o controlo da camada de correlacionamento de alertas e é responsável pelas ações da camada de investigação.
Escolheu-se para o componente web do sistema uma arquitetura Cliente-Servidor de N-camadas, pois possui a estrutura adequada para uma plataforma web, oferece uma boa performance para a baixa quantidade de pedidos que vão ser criados pelos operadores, escalabilidade, modularidade e resistência a falhas graças ao isolamento de cada camada e a capacidade de replicação individual de cada uma sem necessidade de modificações. Juntando a estas vantagens, também já existem frameworks de software prontas para colocar um sistema com este tipo de arquitetura em produção rapidamente. Desta escolha resultaram então as seguintes camadas:

- **Camada de Dados** - Possui as investigações criadas pelo componente de investigações e dados relativos à apresentação das mesmas;
- **Camada de Lógica** - É responsável por buscar os dados da camada de dados, decidir os dados e conteúdos a apresentar, renderizar as vistas, enviar estas ao cliente e mapear feedback do operador para ser enviado à camada de investigação de modo a auxiliar a aprendizagem do classificador do Waldo;
- **Camada de Apresentação** - Tem a interface a apresentar ao operador de modo a poder desfrutar das investigações do Waldo e oferecer feedback quando necessário.

5.3 Desenho e Especificação da Arquitetura Interna

Nesta secção é apresentada toda a arquitetura interna, explicado o funcionamento de cada componente e o porquê da escolha destes.
A partir da solução escolhida para a arquitetura na secção 5.2 decidiram-se então os módulos necessários para conseguir implementar as funcionalidades pretendidas, como se poder ver na figura 5.3.

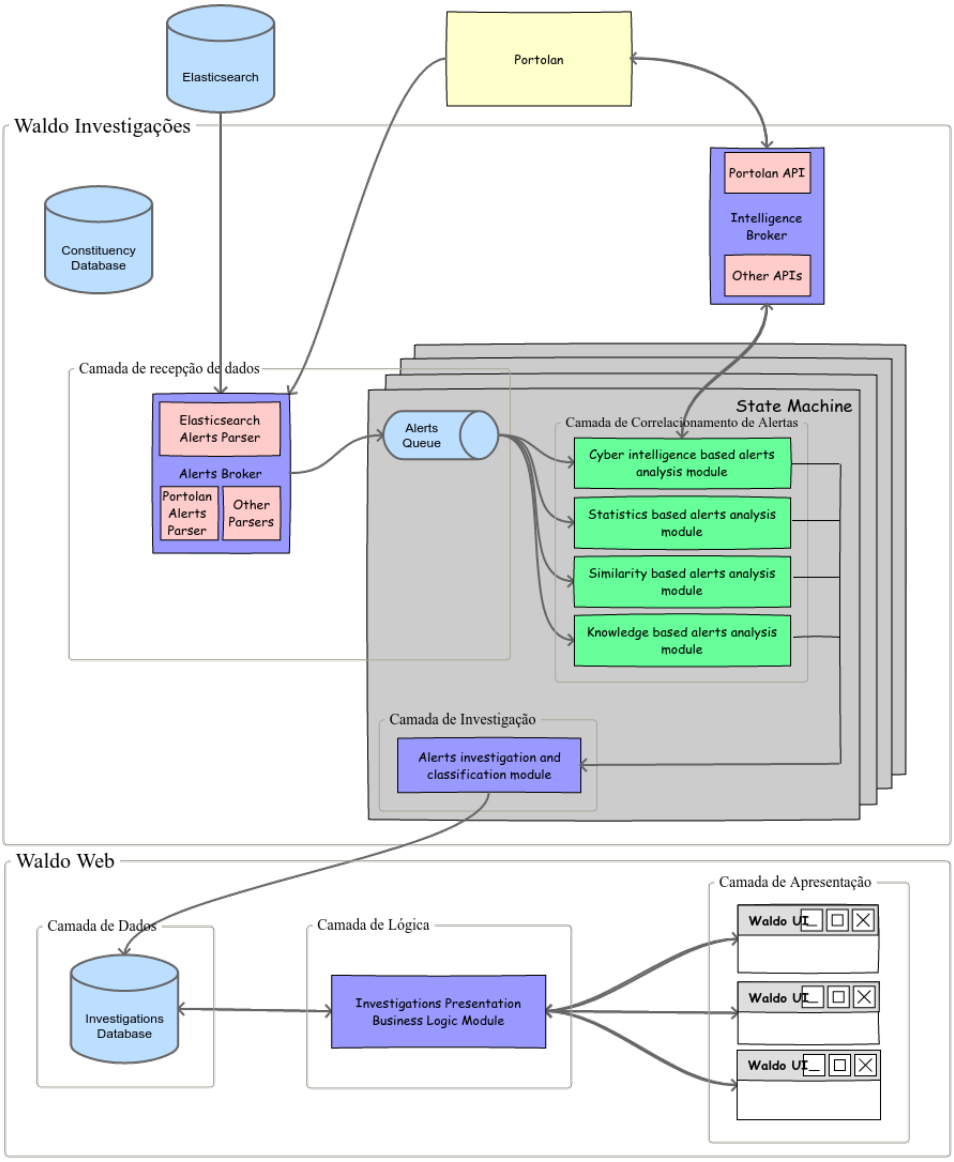


Figura 5.3: Arquitetura Interna do Sistema

Os módulos apresentados têm como objetivo:

- **Alerts Broker** - Responsável pela comunicação com as várias fontes de dados e, após recepção, responsável por validar se os dados recebidos correspondem a mensagens/alertas válidos através da utilização de vários parsers;
  - **Logstash Alerts Parser** - Utilizado para identificar e validar alertas recebidos do logstash da organização;
  - **Portolan Alerts Parser** - Procura identificar os pedidos de reanálise de alertas que a plataforma de inteligência da organização vai enviar quando obtiver novas informação relativas a máquinas suspeitas;
  - **Other Parsers** - Caso haja novas fontes de informação podem ser construidos parsers para permitir a receção dos seus dados.
- **Máquina de Estados** - Visa a gestão do processo de análise de alertas, desde o retirar do alerta da queue, o correlacionamento do alerta nos vários correlacionadores até à classificação do alerta e a criação da investigação. Os correlacionadores realizam uma análise contínua que vai modelando o estado da rede, influenciando análise futuras. Deve então existir uma separação entre as várias redes de clientes de forma a diminuir a possibilidade de erro. Esta separação é realizada construindo uma máquina de estados para a rede de cada cliente com os componentes:
  - **Alerts Queue** - Recebe os alertas que forem validados no Alerts Broker e que sejam para a rede do cliente associado há máquina de estados. Quando a máquina de estados está no estado de busca de alertas recolhe um alerta da queue e passa ao próximo estado;
  - **Cyber Intelligence Based Alerts Analysis Module** - Módulo que procura correlacionar o alerta recebido com dados de fontes de inteligência disponíveis, nesta primeira fase com dados fornecidos pelo Portolan;
  - **Statistics Based Alerts Analysis Module** - Utiliza dados estatísticos da rede continuamente de forma a identificar relações entre os vários alertas de rede e construir possíveis cenários de ataque. Este tipo de correlacionador não é o mais preciso, contudo oferece capacidade de identificação de ataques desconhecidos;
  - **Similarity Based Alerts Analysis Module** - Baseia-se na análise de semelhanças dos vários alertas, normalmente tirando partido de técnicas de clustering de modo a entender os vários grupos de ataques, identificar outros que lhes sejam semelhantes e até identificar grupos de ataques desconhecidos;

– **Knowledge Based Alerts Analysis Module** - Aplica análise de pré-requisitos ou consequências, é a técnica com maior capacidade de acerto. Contudo é necessário um grande conhecimento de domínio e não consegue detetar tipos de ataque desconhecidos;
– **Alerts Investigation and Classification Module** - Quando todos os correlacionadores acabarem a sua análise a State Machine passa do estado de correlação de alertas para o estado de classificação de alertas, em que este módulo é responsável por pesar o resultado de cada um dos classificadores anteriores e atribuir uma classificação final de ataque ou não. Ao mesmo tempo que é responsável por atribuir a classificação deve juntar os dados e criar uma investigação para visualização por parte do operador e para futuro treino. Após este processo a State Machine volta ao estado de busca de alertas.

O fluxo da máquina de estados segmenta-se nos seguintes estados e pode ser visualizado na figura 5.4.

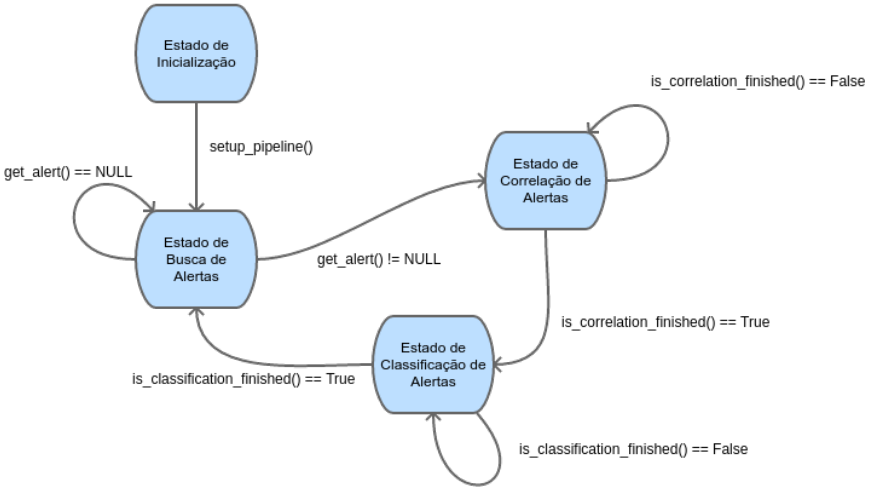


Figura 5.4: Máquina de Estados de Correlacionamento de Alertas

– **Estado de Inicialização** - Quando é construida a máquina de estados devem ser construidos os vários elementos desta. Quando terminada a inicialização destes passa ao estado de Busca de Alertas;
– **Estado de Busca de Alertas** - Se existirem alertas na queue e a máquina estiver neste estado, um alerta é recolhido para análise e passa-se ao estado de correlação de alertas;

  - **Estado de Correlação de Alertas** - Os vários correlacionadores são ativados e realizam uma análise ao alerta recebido consoante os seus estados internos. Quando todos os correlacionadores terminarem esta tarefa são recolhidos os resultados e passa-se ao estado de classificação de alertas;
  - **Estado de Classificação de Alertas** - Recebe os dados dos vários correlacionadores e pesa estes utilizando o classificador associado ao respetivo correlacionador, em caso de existir um bom nível de suspeita é criada uma investigação que vai ser registada na base de dados de investigações com os alertas relacionados, voltando-se finalmente ao estado de inicialização.

- **Intelligence Broker** - Responsável pela comunicação com as plataformas de inteligência disponíveis. Deve abstrair a comunicação com os vários módulos de inteligência e ao mesmo tempo devolver os dados de uma forma estruturada para serem utilizados pelo módulo de correlacionamento de cyber intelligence;
  - **Portolan API** - API para saber como realizar a comunicação com o Portolan;
  - **Other APIs** - API para comunicação com as fontes de inteligência existentes.
- **Constituency Database** - Base de dados com a constituição da rede de cada cliente de modo a saber quantas máquinas de estados é preciso criar;
- **Investigations Database** - A base de dados de investigações é utilizada para registar as investigações realizadas pelas várias State Machines, para buscar dados para apresentar aos operadores na dashboard e para após receber feedback permitir o treino do módulo de investigação e classificação;
- **Investigations Presentation Business Logic Module** - Módulo responsável por fazer a ponte entre o modelo de dados e a camada de visualização. Possui a lógica dos conteúdos a serem enviados ao cliente e a resposta às várias interações;
- **Waldo UI** - São as páginas apresentadas ao utilizador de modo a conseguir entender os dados apresentados e utilizar as funcionalidades da aplicação.

5.4 Ferramentas e Tecnologias Adotadas

Nesta secção são apresentadas as ferramentas e tecnologias que vão ser utilizadas no desenvolvimento deste projeto.

5.4.1 Criação de Mockups

Para a construção dos modelos, figuras e mockups do relatório será utilizado o Pencil[39]. Esta é uma ferramenta cross-platform que foi construida com o intuito de prover uma ferramenta livre e open-source para prototipar nas plataformas mais populares.

5.4.2 IDS

De modo a ser possível a criação de todo o pipeline para testar o sistema, vão ser utilizados os IDSs apresentados na seção 5.1.

5.4.3 Sistema Operativo

De modo a ser possível a utilização dos IDSs referidos no ponto anterior é necessário um sistema a que todos sejam compativeis. Para tal é necessário um sistema operativo baseado em GNU/Linux[11]. O standard na empresa é a utilização da distribuição Debian[40] de GNU/Linux. Esta distribuição tem bastante cuidado com a qualidade de todos os software existentes nos seus repositórios realizando grandes períodos de testes antes de cada versão estável. Vai permitir tanto o bom funcionamento e interligação entre todos os IDSs apontados como a compatibilidade com todas as distribuições debian based revelando-se como uma escolha bastante pertinente.
Será utilizado também o sistema operativo Kali[41] para vários dos testes a serem aplicados. Este é baseado no sistema Debian, contudo possui uma grande quantidade de ferramentas pré-instaladas e configurações adaptadas especificamente para realizar testes de intrusão.

5.4.4 Servidor Web

Nginx[42] (pronunciado "engine-x") é um servidor HTTP e proxy reverso, bem como um servidor de proxy de email e um servidor proxy TCP genérico, originalmente escrito por Igor Sysoev. Por longos períodos temporais tem sido utilizado por sites russos extremamente pesados incluindo Yandex, Mail.Ru, VK, e Rambler.
De acordo com Netcraft[43], nginx serviu 21.43% dos sites mais ocupados em abril de 2015.
O Nginx é rápido e leve como demonstrado por Barry[44]. Também tem outras vantagens como por exemplo ser de fácil configuração.
Outra possibilidade seria adotar o Apache, o servidor web mais utilizado. No entanto, como o servidor Nginx é mais rápido a servir conteúdos estáticos, consome muito menos memória ao receber pedidos concorrentes e é fácil de configurar como indicado no artigo escrito por *Will Reese*[45], o autor considerou que se enquadraria melhor e decidiu utilizá-lo.

5.4.5 Base de Dados

Postgres[46] é um sistema de gestão de base dados objeto-relacional (ORDBMS), com ênfase na capacidade de extensão e padrões de conformidade. Como servidor de base de dados, a sua principal função é armazenar informação de forma segura, apoiando-se nas melhores práticas, e permitindo a recuperação da mesma a pedido de outras aplicações.
O PostgreSQL está de acordo com o principio ACID, sendo este bastante reconhecido por possuir bases mais rigorosas na sua aproximação de robustez e integridade de dados.
O autor ainda considerou a base de dados mySQL, mas com base na comparação de Tezer[47] averiguou-se que o PostgreSQL oferece melhores condições em termos de robustez e de segurança.

5.4.6 Broker

O principal ponto para os *brokers* é a forma como vão receber e passar a informação, para tal foram consideradas as soluções:

- **Redis** - foca-se bastante em performance o que é bom, contudo para tal abdica da ordem das queues que para o caso de uso a ser aplicado é crucial;
- **Kafka** - Apresenta-se como uma solução altamente escalável e para ser utilizada e distribuída por clusters. Pode existir a necessidade de replicar o broker de comunicação com fontes, contudo o que está a ser recebido são alertas dos vários IDS e não eventos raw, o que cria uma baixa quantidade de entradas e não justifica um sistema desta envergadura;
- **RabbitMQ** - Permite a fácil integração dos seus clientes com outros sistemas e com a estrutura pretendida para o sistema.

Foi escolhido o RabbitMQ pois adequa-se ao fluxo de eventos necessário e é simples de utilizar. Este será utilizado com o seu cliente de Java[48] para a comunicação na parte de investigações e com o cliente Pika[49] em python para interagir com as suas queues na parte web.

5.4.7 Tratamento de Eventos no Cliente

jQuery[50] - Será utilizado pois facilita no tratamento de eventos, manipulação das páginas, compatibilidade entre vários browsers, na utilização de Ajax[51] e é bastante utilizado na industria de modo que é estável, bastante fiável, já conta com uma grande quantidade de exemplos e podem-lhe ser associados plugins que estendem as suas funcionalidades.

5.4.8 Representação das Investigações do Waldo

Para fazer a representação das investigações do Waldo vai ser utilizada uma das seguintes bibliotecas de código:

- **Arbor.js**[52] - É pequena, permite a demonstração das ligações entre elementos, é simples de utilizar e não necessita de dependências nenhumas extra visto que o Waldo Web já vai utilizar jquery;
- **Treant.js**[53] - Depende da biblioteca de código raphael.js[54] que simplifica a criação de gráficos vetoriais na web, contudo desta mapeia apenas funcionalidades para os grafos de árvore que são o foco do Treant.js, ficando bloqueada a utilização de várias funcionalidades existentes no raphael.js que poderiam ser úteis. A biblioteca já possui uma maior complexidade na construção de grafos em relação ao Arbor.js, contudo também conta com uma boa quantidade de exemplos das várias funcionalidades o que permite também uma simples construção destes grafos de árvore;
- **Vis.js**[55] - Esta possui vários módulos, um com o core e depois cada um dos outros possui conjuntos de funcionalidades. Para o Waldo Web podemos utilizar apenas o core e o módulo de redes. Mesmo com esta separação cada um destes possui um tamanho considerável e pode representar já algum peso no tempo de carregamento da página e na construção dos grafos. Esta é a que apresenta uma maior quantidade de funcionalidades e que possui uma melhor adaptação a diferentes casos de uso. A complexidade de construção de grafos é superior ao Arbor.js e menor que o Treant.js, esta biblioteca é a que conta com maior quantidade de documentação e exemplos.

5.4.9 Parse de Alertas JSON

Para ler os alertas recebidos na queue externa será necessário um parser de JSON para java, para tal consideraram-se as seguintes bibliotecas:

- **json-simple**[56] - É fácil de utilizar, possui todas as funcionalidades pretendidas para o parse de JSON do Waldo e ao mesmo tempo é bastante pequena;
- **GSON**[57] - É rápido, é suportado pela google então qualquer bug que apareça será corrigido rapidamente e a biblioteca apresenta uma qualidade da codebase alta.

Analisando os resultados apresentados por *Josh Dreyfuss* em [58], foi escolhida a json-simple pois mesmo ficando em segundo em termos de velocidade a diferença foi baixa relação ao GSON, e considerando o seu tamanho apresenta uma maior confiança de que não será um overhead para o sistema.

5.4.10 Linguagens

Para responder às várias atividades e módulos a desenvolver vão-se utilizar as seguintes linguagens de programação:

- **Python**[59] - Utilizada para programar a lógica do lado do Waldo web e tratar dados para visualização em R.
- **html**[60] - Utilizada para os templates do Waldo Web.
- **css**[61] - Utilizada para os templates do Waldo Web.
- **js**[62] - Utilizada para os templates do Waldo Web.
- **postgreSQL**[46] - Linguagem utilizada para manipulação das bases de dados postgres.
- **JSON**[63] - Utilizada para transferências de dados de forma estruturada.
- **Bourne Shell(Bash)**[64] - Utilizada em vários casos para automatização e integração dos vários sistemas no ambiente de desenvolvimento.
- **LaTeX**[65] - Utilizada para escrever a dissertação.
- **R**[66] - Utilizada para análise estatística e visualização dos dados das várias features e labels.
- **Java**[67] - Utilizada para construir e integrar o sistema de Waldo investigações.

5.4.11 Frameworks

Django[68] - Utilizada para construir rapidamente e de forma segura componentes web, ajudando na base de dados, na lógica e na simplificação da criação das páginas de visualização. Esta utiliza o padrão MTV(model - template - view)[69] em que:

- O model possui a descrição dos modelos de dados existentes;
- O template são às páginas a serem apresentadas ao utilizador e forma como os dados são apresentados nestas;
- A view é responsável pela resposta aos pedidos do utilizador e por decidir quais os dados a passar aos templates.

Bootstrap[70] - Framework com componentes para melhorar o aspeto da página apresentada ao utilizador.

5.4.12 Controlo de Versões

Git[71] - É um software de controlo de versões para código fonte. Permite a fácil consulta de modificações entre as várias versões, entender quem realizou as modificações, criar diferentes ramos de desenvolvimento de modo a caso várias pessoas estejam a desenvolver as suas modificações não partirem o código do outro, a junção de modificações de várias pessoas no mesmo projeto, entre outros.

5.4.13 Tecnologias de Desenvolvimento

Intellij IDEA[72] - É um IDE de java que possui análise inteligente do código do contexto em que estamos a desenvolver, permitindo um desenvolvimento rápido e simultaneamente menos propicio a erros.
Pycharm[73] - É um IDE de python que possui análise inteligente do código do contexto em que estamos a desenvolver, permitindo um desenvolvimento rápido e simultaneamente menos propicio a erros.
Sublime Text[74] - É um editor de texto que possui funcionalidades como snippets que permitem a criação de templates associados a keywords, macros que executam ações automatizadas, esquemas de cores para uma grande variedade de linguagens, e permite a extensão das suas funcionalidades de forma bastante simples por qualquer pessoa, tendo várias funcionalidades extra criadas pela comunidade em volta. Vai ser utilizado na programação de HTML, CSS, JavaScript, Bash e outras linguagens que possam ser necessárias.
GNU nano[75] - É um editor de texto de terminal. Vai ser utilizado para editar ficheiros de configuração e para modificações a ficheiros realizadas no terminal.

5.4.14 Validação de Funcionalidades

JUnit[76] - É uma framework para escrever testes. Simplifica a automatização de tarefas nos testes e a gestão do ciclo de vida dos mesmos;
Intellij IDEA[72] - Além das capacidades apresentadas na secção 5.4.13, este possui capacidade de análises estáticas de código, e apresenta métricas de coverage;
unittest[77] - Framework para auxiliar na escrita, execução e gestão do ciclo de vida dos testes unitários. Esta é uma ferramenta standard em python e também é utilizada de forma integrada pela framework Django;
Graph Coverage[78] - É uma plataforma que recebendo os nós e as ligações de um fluxo de código gera um modelo de grafo consoante o tipo de coverage e algoritmos selecionados;
GNOME Clocks[79] - É uma ferramenta que possui funcionalidade de temporizador, cronometro, alarme e visualização de vários relógios do mundo.

5.5 Conclusão

Com este capitulo de arquitetura do sistema conseguimos entender em que ponto se enquadra o Waldo, como vai interagir com os outros componentes existentes, como é constituído e de que forma vai executar o que é pretendido. Definiu-se também as tecnologias necessárias e ferramentas à sua construção.
Este planeamento apresenta-se como uma grande mais valia porque vai enriquecer e validar as capacidades da plataforma e ajudar a uma implementação mais cuidada e com um foco preciso.

Capítulo 6

Metodologia de Desenvolvimento

Neste capitulo é descrita a metodologia escolhida para o desenvolvimento do projecto.

6.1 Pontos a considerar

Para a seleção de uma metodologia é necessário ter em conta cada um dos seguintes pontos:

- **Tamanho do projeto** - Visto que o projeto tem a duração limitada ao tempo do estágio curricular e que se pretende implementar uma base experimental, este pode-se definir como tendo uma dimensão média;
- **Criticalidade** - Com este projeto pretende-se apresentar um protótipo experimental de tecnologias inteligentes que permita o melhoramento da deteção e resposta a incidentes, como tal, este não é um sistema critico;
- **Time-to-market** - Não existem ferramentas no mercado que permitam cumprir com todos os requisitos pretendidos, desta forma um rápido lançamento do produto completo seria ideal;
- **Orçamento** - Para o projeto possui-se um orçamento de 1200 horas e o material necessário para desenvolvimento e simulação;
- **Estabilidade de requisitos** - A visão principal do projeto não vai mudar, contudo após a implementação de cada um dos módulos de análise e correlacionamento podem ser encontrados aspetos interessantes que influenciem os próximos passos a tomar. Desta forma os requisitos são instáveis;

- **Tamanho da equipa** - A equipa de requisitos, design, implementação, validação e manutenção do Waldo possui um membro a tempo inteiro, o autor do relatório, e esporadicamente os orientadores da Dogædis e do ISEC para aconselhamento, caso necessário;
- **Experiência dos developers** - O autor está a terminar o seu mestrado em Informática e Sistemas, possui dois anos de experiência de desenvolvimento de software profissionalmente e já construiu vários projetos de sistemas inteligentes nos seus tempos livres. Sendo como tal uma equipa com experiência média;
- **Cultura de gestão e organizacional** - A organização já possui processos de desenvolvimento bem definidos.

6.2 Escolha da metodologia

Após analise dos processos de R&D da organização e a partir das vantagens e desvantagens apresentadas em [80] e [81] sobre metodologias ágeis e tradicionais decidiu-se a adoção de uma metodologia baseada numa abordagem ágil com as seguintes características:

- **Reuniões a cada sprint** - De modo a manter a visão e o acompanhamento do projeto consistente foram realizadas reuniões a cada sprint com o orientador da empresa. Foram também realizadas reuniões com o orientador do ISEC com frequência mensal, para acompanhamento do projeto e esclarecimento de dúvidas;
- **Desenvolvimento Iterativo** - Visto que é um projeto arriscado e experimental, os resultados podem ir-se afastando do previsto ao longo do tempo e não decorrer como planeado, desta forma a melhor abordagem será desenvolver realizando releases pequenas e frequentes para obter feedback constante do cliente conseguir responder priorizando as necessidades do cliente. Cada sprint vai ter entre uma a quatro semanas, sendo que inicialmente os sprints serão mais curtos, visto que as tarefas de maior risco serão realizadas mais cedo e aí o acompanhamento será mais importante. Ao longo do projeto, e à medida que o risco das tarefas diminua a duração de cada sprint irá aproximando-se das quatro semanas;
- **Continuous Integration** - Controlo de versões com integração continua é utilizado para trabalhar sempre na versão mais recente, permitir voltar a uma versão mais antiga caso haja algum problema e para permitir a consulta do código por quaisquer partes interessadas a qualquer momento;

- **Coding Standards** - De modo a simplificar a partilha do código e garantir que este é consistente ao longo do projecto serão utilizados:
  - Os coding standards da Apache[82] para a implementação em Java;
  - O PEP8[83] para a implementação em Python.

6.3 Ciclo de vida

O panorama geral do ciclo de vida pode ser observado na figura 6.1.

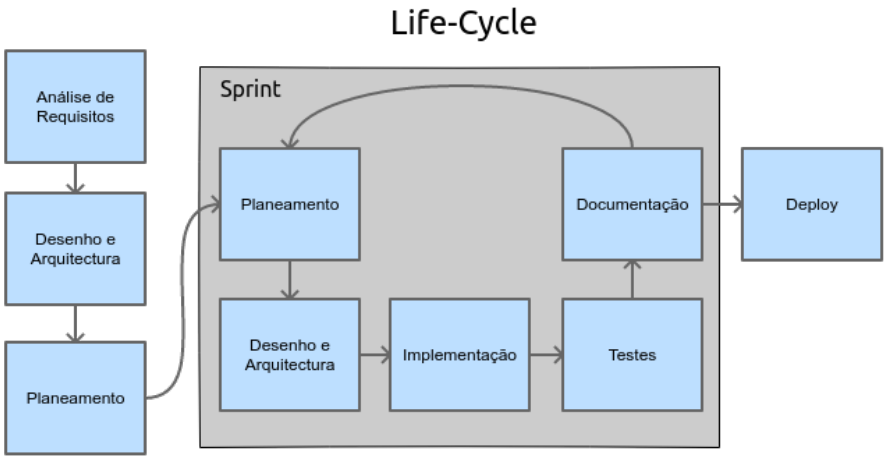


Figura 6.1: Life-Cycle

6.3.1 Análise de Requisitos

Inputs

- Proposta de estágio.

Tarefas

- Análise das tarefas e objetivos propostos pelo cliente e conversão das suas descrições para o formato proposto pelo EARS;
- Reuniões com o cliente e partes interessadas de modo a entender a visão deste e garantir que tudo o que é necessário foi levantado;
- Escrita de requisitos funcionais e não funcionais;
- Escrita de user stories com base nas tarefas e objetivos.

Outputs

- Requisitos Funcionais;
- Requisitos Não Funcionais;
- User stories definidas.

6.3.2 Desenho e Arquitetura

Inputs

- Requisitos funcionais;
- Requisitos não funcionais;
- User Stories.

Tarefas

- Especificação da estrutura do sistema;
- Decisão de ferramentas a utilizar em cada módulo;
- Explicação do modo como vão ser abordados os requisitos a implementar;
- Escolha e escrita de ferramentas a utilizar;
- Escrita do desenho dos vários módulos a construir e utilizar;
- Adição de tarefas de implementação a user stories existentes e criação de novas user stories;

Outputs

- User Stories com tarefas de implementação associadas;
- Estrutura do sistema a implementar;
- Escolha de Ferramentas a utilizar.

6.3.3 Planeamento

Inputs

- User stories.

Tarefas

- Estimar cada uma das user stories;
- Ordenar as user stories por prioridade do cliente;
- Definir milestones;
- Perceber principais riscos que possam afetar o sucesso das milestones e pensar como os mitigar;
- Construir um backlog com as user stories ordenadas e categorizadas por milestones.

Outputs

- Riscos associados;
- Product backlog.

6.3.4 Desenvolvimento

O desenvolvimento será iterativo e constituído pelas etapas:

Planeamento

- Inputs
  - Product backlog;
  - Ultimo sprint backlog.
- Tarefas
  - Passar tarefas ainda não terminadas do backlog do ultimo sprint para o novo sprint backlog;
  - Reunião com o cliente para apresentar as ultimas modificações;
  - Reajustar estimativas caso necessário;
  - Adicionar/reajustar user stories caso se tenha percebido coisas em falta com os últimos desenvolvimentos ou consoante o feedback do cliente;
  - Escolher user stories prioritárias do product backlog e consoante as estimativas colocar no novo sprint backlog.
- Outputs
  - Novo Sprint backlog.

Desenho e Arquitetura

- Inputs
  - Sprint backlog.
- Tarefas
  - Planear implementação de novos módulos resultantes do feedback do cliente;
  - Reajustar desenho e arquitetura de módulos já existentes caso necessário;
  - Associar tarefas de implementação a novas user stories criadas.
- Outputs
  - Estrutura do módulos a implementar;
  - Escolha de Ferramentas a utilizar;
  - Sprint backlog.

Implementação

- Inputs
  - Sprint backlog.
- Tarefas
  - Implementar tarefas descritas nas user stories;
  - Atualizar estados das user stories do sprint backlog;
  - Integrar novos módulos no sistema já existente;
  - Descrição na user story de ações tomadas para resolver cada tarefa;
  - Controlo de versões;
  - Debug.
- Outputs
  - Sprint backlog;
  - Módulos Implementados.

Testes

- Inputs
  - Sprint backlog;
  - Módulos Implementados.
- Tarefas
  - Implementação de testes para os novos módulos implementados;
  - Execução dos testes de regressão;
  - Mudar estado de user stories para "DONE" caso implementação passe nos testes, ou manter em "IN PROGRESS" e adicionar feedback na user story do caso de teste que não passe.
- Outputs
  - Resultados dos testes;
  - Testes implementados;
  - Sprint backlog.

Documentação

- Inputs
  - Resultados dos testes;
  - Sprint backlog.
- Tarefas
  - Escrever dificuldades encontradas e as soluções escolhidas para ultrapassar essas dificuldades;
  - Escrever explicação para o funcionamento de cada um dos módulos do sistema;
  - Escrever manuais para a expansão e compreensão de novos passos a tomar;
  - Documentar bases de conhecimento encontradas para a construção do sistema;
  - Documentar resultados de testes.
- Outputs
  - Dissertação de Mestrado;
  - Documentos de Apoio;
  - Sprint backlog de fim de sprint;
  - Módulo completo.

6.3.5 Deploy

Inputs

- Sistema Mais Recente;
- Documentos de Apoio.

Tarefas

- Configurar queues externa de rabbitMQ;
- Configurar queues de rabbitMQ de cada um dos clientes;
- Configurar e ativar servidor web;
- Configurar base de dados postgreSQL;
- Executar os scripts de geração do modelo de dados do Waldo Web;
- Arranque do sistema Waldo Investigações;
- Ativar o sistema Waldo Web.

Outputs

- Sistema em produção.

6.4 Conclusão

Neste capitulo foi apresentada a metodologia adotada para cada uma das etapas de desenvolvimento do projeto. Tiveram-se em conta as características do projeto e a partir dai escolheu-se a forma mais adequada de trabalhar. Agora já com uma metodologia de trabalho bem definida, vamos no próximo capitulo passar à implementação do sistema.

Capítulo 7

Implementação

Neste capítulo é descrita a implementação da plataforma com base nos requisitos funcionais, na proposta arquitectural, nas ferramentas propostas e no planeamento.

7.1 Receber Logs

Para receber logs do exterior foi:

- Configurado e colocado em execução um servidor de RabbitMQ;
- Criada uma queue para receber informação do exterior;
- Criado um consumer em java utilizando o cliente de java do RabbitMQ para responder a cada um dos eventos recebidos na queue exterior, sendo este consumer responsável por:
  - Realizar o parsing dos alertas recebidos com o formato JSON;
  - Ao observar a ferramenta que gerou o alerta atribuir um tipo ao mesmo;
  - Reencaminhar para a queue de alertas do respetivo cliente.

A implementação do consumer divide-se em:

1. **Parsing de Alertas** - Para o parsing do consumer foi utilizada a libraria json-simple de java, interpretando os alertas recebidos em json e transformando-os em objetos da classe JSONObject que possuem as chaves e valores de cada atributo do alerta;
2. **Atribuir tipo ao Alerta** - Para atribuir um tipo ao alerta foi implementada a classe AlertFactory que é responsável por receber um objeto JSONObject, reconhecer o tipo de alerta consoante a ferramento que o gerou e devolver um objecto filho da classe Alert associado ao tipo identificado. Os tipos de alertas existentes são:

- **Alert** - Classe abstrata que representa a informação base para um alerta e o seu funcionamento;
- **BroAlert** - Classe que representa os alertas que provem do IDS BRO e os seus dados específicos;
- **OssecAlert** - Classe que representa os alertas que provenientes do IDS OSSEC e os seus dados específicos;
- **SnortAlert** - Classe que representa os alertas que com origem no IDS Snort e os seus dados específicos;
- **PortolanAlert** - Classe que representa os avisos que podem ser enviados pela plataforma de ciber inteligência da empresa.

Pode ver-se a representação destes na figura 7.1.

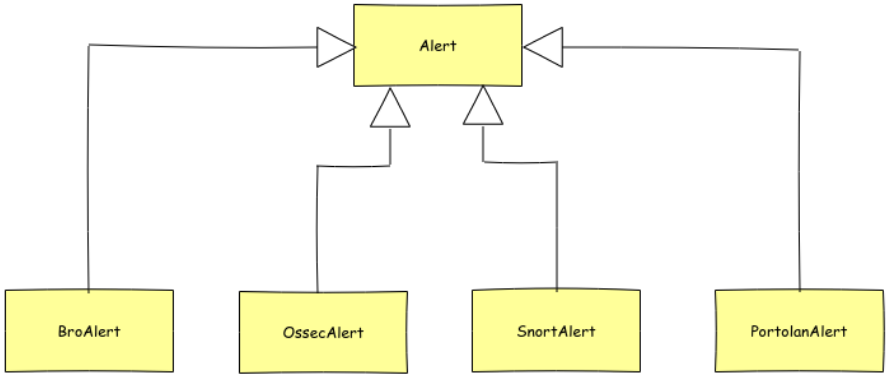


Figura 7.1: Tipos de Alertas

- **3º Enviar alerta para cliente** - Foi criada a class BrokerProducer que é um producer de Alertas. Este é responsável pela serialização dos objetos Alert resultantes e pelo envio destes para as queues dos clientes a que estão associados para posterior análise.

O fluxo global deste processo pode ser observado na figura 7.2.

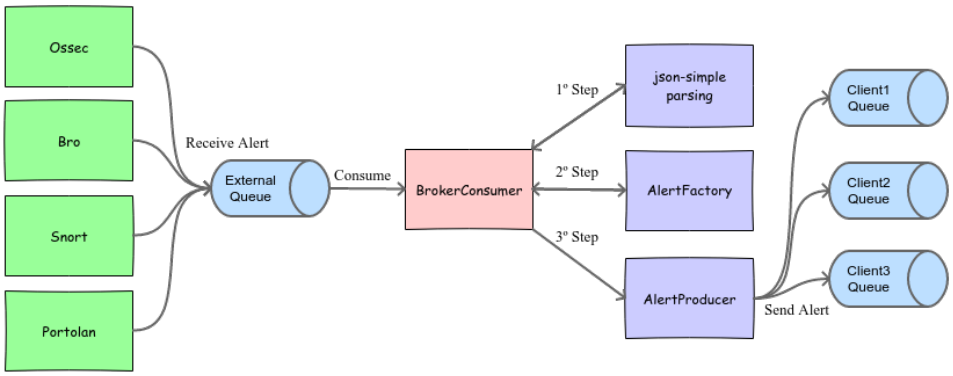


Figura 7.2: Pipeline Receção de Alertas

7.1.1 Adaptabilidade a várias fontes diferentes

Para registar novas fontes de informação no sistema basta realizar os passos:

- Enviar outputs da nova fonte ao logstash que recebe os alertas dos IDS;
- Implementar classe de alerta associada à nova fonte, esta deve estender a classe pai Alert que já possui os parâmetros principais do parse e gestão dos alertas;
- Adicionar o novo tipo de alerta na enumeração de AlertType;
- Adicionar tipo e chamada para construção do novo objeto de alerta no método getAlert da AlertFactory.

7.2 Máquina de Estados

Para a máquina de estados foi criada a classe StateMachine que é responsável por gerir o fluxo da análise de alertas. Este fluxo foi construido a partir da definição dos 4 estados apresentados na figura 5.4. Foi criada a interface State que define os métodos obrigatórios para que uma classe seja um estado da StateMachine, sendo implementada por todas as classes de estados que estão associadas aos referidos anteriormente, como se pode ver na figura 7.3.

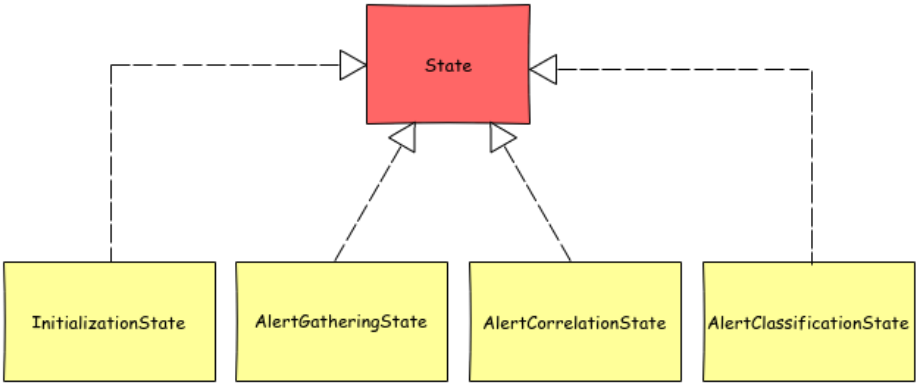


Figura 7.3: Estados da Máquina de Estados

Para realizar o fluxo necessário à análise de alertas cada estado realiza vários passos:

- **InitializationState** - Inicializa os correlacionadores e classificadores, passando de seguinda para o estado AlertGatheringsState.
- **AlertGatheringState** - Se existirem alertas na queue do cliente consome um alerta e passa ao próximo estado, se não existirem alertas espera;
- **AlertCorrelationState** - Ativa os correlacionadores e espera até que todos indiquem que terminaram o seu correlacionamento de alertas. Quando todos terminarem passa para o estado AlertClassificationState;
- **AlertClassificationState** - Ativa os classificadores e espera até que todos indiquem que terminaram a classificação de alertas. Cada um destes avalia os resultados obtidos dos correlacionadores, e se for detetada uma classificação suspeita são registados os resultados como uma investigação na base de dados da plataforma web do Waldo. Ao concluir estes passos com sucesso passa a execução para o estado AlertsGatheringState.

7.2.1 Modularidade

Durante o estado AlertCorrelationState serão ativados todos os correlacionadores registados no estado InitializationState e durante o AlertClassificationState serão ativados todos os classificadores registados também durante o InitializationState. Cada um destes corre numa thread separada e quando todos terminam passa-se ao estado seguinte.

Para garantir modularidade procurou-se a fácil adição de novos pares correlacionador/classificador. Para tal os vários pares tem de estar totalmente desacoplados. Um classificador pode depender de dados de um correlacionador com que vai emparelhar, contudo não existem dependências entre o correlacionador/classificador de um par e o correlacionador/classificador de outro. Para registar novos módulos é necessário:

- Adicionar um novo tipo à enumeração InvestigationType;
- Construir uma classe de correlacionador que estenda a classe pai Correlator;
- Implementar método abstrato executeTasks() com a lógica necessária à funcionalidade do correlacionador;
- Ao terminar a execução do correlacionador passar resultados para a máquina de estados identificando o tipo de correlacionador;
- Construir uma classe de classificador que estenda a classe pai Classifier;
- Implementar método abstrato executeTasks() com a lógica necessária à funcionalidade do classificador;
- Implementar método abstrato registerPossibleAttack() para registar o modelo do classificador atual associado à investigação a ser registada;
- Implementar método abstrato applyOperatorFeedback() para ser chamado quando for recebido feedback dos operadores na aplicação waldo web;
- Ao terminar a execução do classificador registar investigação com modelos associados na base de dados Waldo Web;
- Criar representação visual dos novos modelos no template de investigação e respetivo mapeamento para os dados e pedidos a utilizar.

7.3 Analisar Dados

Para realizar a análise dos dados serão feitos correlacionamentos entre os alertas recebidos aplicando algoritmos baseados em estatística. Para definir a estrutura base de cada um destes correlacionadores foi implementada a classe abstrata Correlator que define cada correlacionador como derivada da classe Thread, possui o registo de alertas que correlacionou e uma funcionalidade para a máquina de estados saber quando este já terminou a sua análise.

7.3.1 Statistics Based Alerts Analysis Module

O correlacionador baseado em estatística, classe StatisticsCorrelator, foi construido com base no artigo apresentado na secção 3.2.5. Este correlacionador baseia-se no conceito de que se um acontecimento no passado teve um certo comportamento é muito provável que este comportamento se repita, desta forma foca-se em criar um estado do global e comparar com o estado de uma janela temporal limitada em procura de mudanças suspeitas. Para tal e com foco no nosso problema, este busca correlacionar os nossos alertas realizando os seguintes passos:

1. **Criação de Hiper Alertas** - Dado que podem ser recebidos alertas semelhantes seguidos em grande quantidade, estes vão afetar as probabilidades de cada tipo de alerta. Para resolver este problema foi pensada a criação de hiper alertas. Para criação destes são associados alertas seguidos que sejam provenientes do mesmo IDS e que possuam vários tipos de dados semelhantes, como se é interno ou externo à rede, o tipo de porto a que está associado, a categoria de alerta, entre outros. Na figura 7.4 pode ser visto um exemplo;

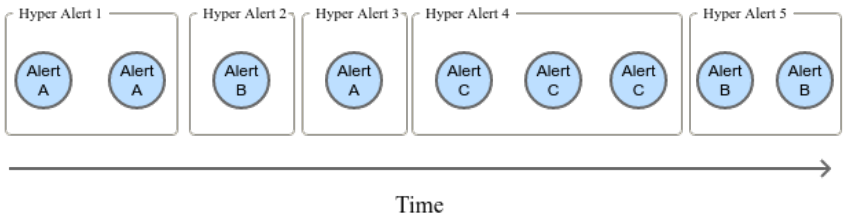


Figura 7.4: Criação de Hiper Alertas

2. **Separação de dados gerais e da janela temporal atual** - Ao serem recolhidos os hiper alertas são registados na janela temporal atual(1 hora por exemplo) e no conjunto total de hiper alertas como exemplificado na figura 7.5, de modo a ser possível calcular a probabilidade de uma categoria de alerta acontecer e a probabilidade de uma categoria de alerta acontecer após outra ter acontecido para a totalidade dos alertas, comparando estes com os mesmos valores limitados à janela temporal definida;

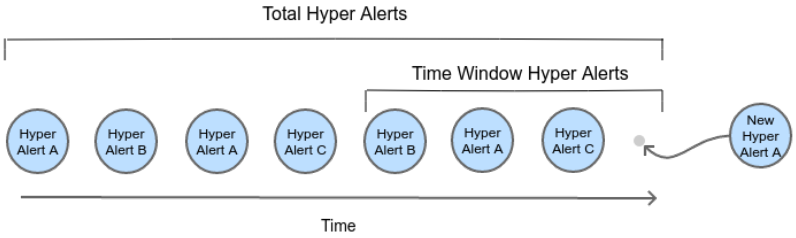


Figura 7.5: Adição de novos Hiper Alertas

3. **Cálculo de Tabela de Correlacionamento** - A partir da ordem dos hiper alertas que são recebidos, estes são associados em pares de alerta seguidos em termos temporais e valida-se a frequência com que estes se repetem, fornecendo desta forma uma indicação do valor de $p(B|A)$ em pares de hiper alertas. Ou seja no par $< A, B >$ entender a probabilidade de acontecer o hiper alerta B dado que o A aconteceu. Deste cálculo são também registados os atributos de A que mais regularmente acontecem nos vários pares de modo a entender quais os atributos que podem ser mais relevantes. Pode-se ver um exemplo do processo na figura 7.6 e um exemplo da tabela resultante na figura 7.7;

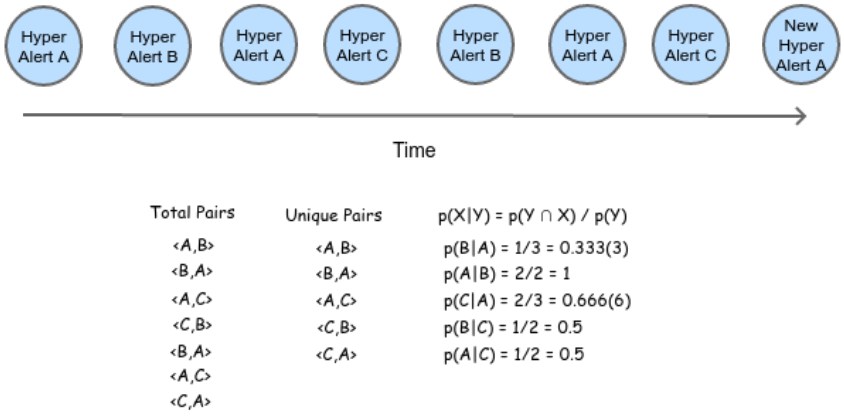


Figura 7.6: Cálculo de Correlacionamento de Hiper Alertas

Alert Pair	Probability	Relevant Atributtes
<A,B>	33.3%	src_ip, dest_ip
<B,A>	100%	src_ip, port
<A,C>	66.6%	timestamp, dest_ip
<C,B>	50%	country_code, src_ip
<C,A>	50%	host, port

Figura 7.7: Tabela de Correlacionamento

4. **Cálculo da Tabela de Relevância** - Com os hiper alertas já existentes esta tabela aglomera a probabilidade de ocorrer cada hiper alerta baseado na sua frequência, apresenta também os tipos de hiper alerta que são mais relevantes para a probabilidade deste acontecer e os que são menos relevantes com base na tabela de correlacionamento. Desta forma permite entender rapidamente quais os tipos de hiper alerta que foram responsáveis por mudanças nas probabilidades de outros hiper alertas. Um exemplo deste cálculo pode ser observado na figura 7.8 e da tabela de relevância na figura 7.9;

Alert Category	Probability
A	p(A) = 4/8 = 0.5
B	p(B) = 2/8 = 0.25
C	p(C) = 2/8 = 0.25

Correlations	Relevance	Result
p(B|A) = 0.333(3)	0.333(3) < RelevanceThreshold	Weakly Relevant
p(A|B) = 1	1 > RelevanceThreshold	Strongly Relevant
p(C|A) = 0.666(6)	0.666(6) > RelevanceThreshold	Strongly Relevant
p(B|C) = 0.5	0.5 > RelevanceThreshold	Strongly Relevant
p(A|C) = 0.5	0.5 > RelevanceThreshold	Strongly Relevant

RelevanceThreshold = 0.4

Figura 7.8: Cálculo de Relevância de Hiper Alertas

Alert Category	Probability	Strongly Related	Weakly Related
A	50%	B, C	---
B	25%	C	A
C	25%	A	---

Figura 7.9: Tabela de Relevância

5. **Cálculo de Probabilidades da Janela Temporal** - Para entender se existem variações significativas nos alertas gerados na última janela temporal, são calculadas as probabilidades de cada tipo de hiper alerta presente no espaço de tempo desta e são mantidos temporariamente;

6. **Deteção de Comportamento Suspeito** - Se existir uma grande variação(consoante o limite definido) da probabilidade de acontecer um hiper alerta que esteja presente nas probabilidades da janela temporal atual em relação ao que está presente na tabela de correlacionamento, é inicializada a análise de um comportamento suspeito na categoria de hiper alerta que despoletou esta análise. Se houver um grande aumento na probabilidade do hiper alerta serão analisados os hiper alertas que são menos relevantes para a probabilidade deste acontecer pois esses são os que mais provavelmente variaram para gerar esse aumento, enquanto se houver uma grande diminuição da probabilidade serão analisados os com maior relevância para este acontecer pois para a probabilidade descer quase certamente que foi uma variação nestes. Se a probabilidade do correlacionamento entre estes tiver uma variação elevada(consoante o limite definido) então este correlacionamento é registado numa tabela de correlacionamento temporário e é definido o caso como corretamente suspeito. Uma demonstração é apresentada na figura 7.10. A demonstração apresenta na primeira etapa a tabela de relevância global com três categorias de alerta e os seus respetivos dados, a probabilidade de relevância limitada apenas à janela temporal atual e o cálculo da variação da relevância entre a tabela da janela temporal actual e a tabela global de relevância. Como a variação foi mais baixa que o limite definido foi despoletada a segunda etapa de análise.
   Na segunda etapa é apresentada a tabela de correlacionamento de global, a probabilidade de correlacionamento associada à janela temporal atual e o calculo da variação da probabilidade entre as mesmas.

Visto que a variação da probabilidade de correlacionamento foi mais baixa que o limite definido foi então despoletado o registo do par de alertas correlacionados que tiveram uma grande variação na tabela temporal de correlacionamento. Posteriormente os elementos desta tabela vão ser utilizados para a criação de cenários de ataque;

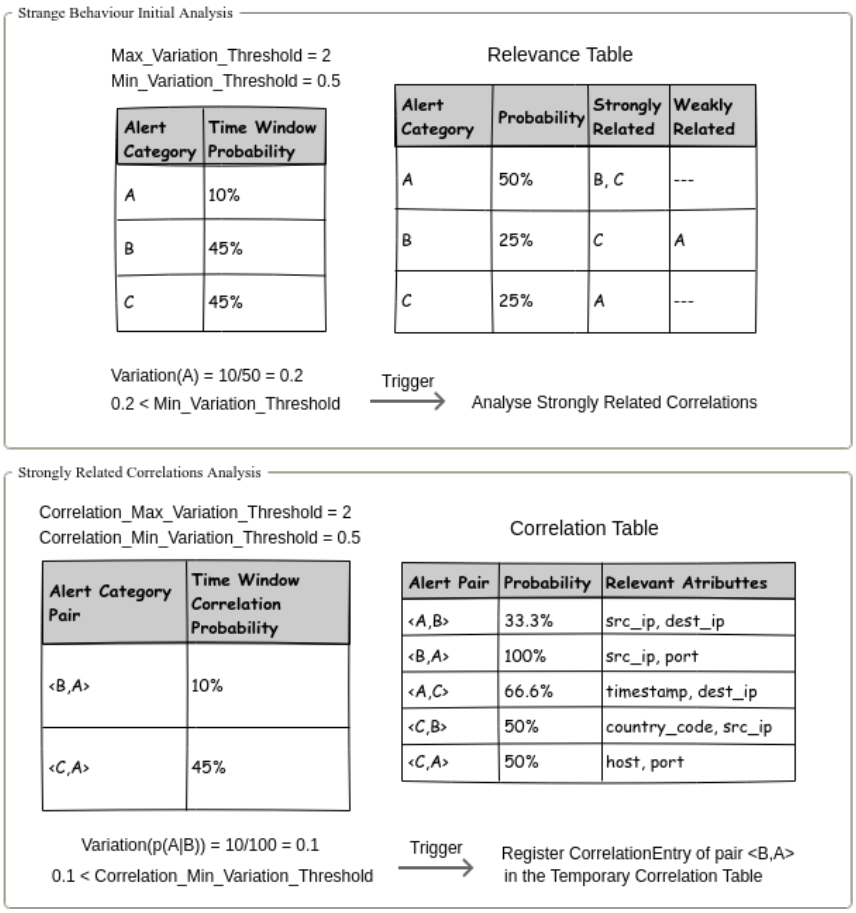


Figura 7.10: Deteção de Comportamento Suspeito

7. **Construção do Cenário de Ataque** - Quando é corretamente classificado um hiper alerta como suspeito é de seguida realizada a construção de um cenário de ataque. A partir da tabela de correlacionamento temporário valida-se se ambos os nós do par registado anteriormente não existem em nenhum dos casos de ataque já definidos. Se não existir é criado um novo cenário de ataque com os nós presentes na tabela de correlacionamento temporário e a sua probabilidade. São

também criados nós para o correlacionamento com tipos de hiper alertas relevantes para o tipo de hiper alerta que despoletou a investigação utilizando as probabilidades da tabela de correlacionamento. Pode-se ver um exemplo deste passo na figura 7.11.

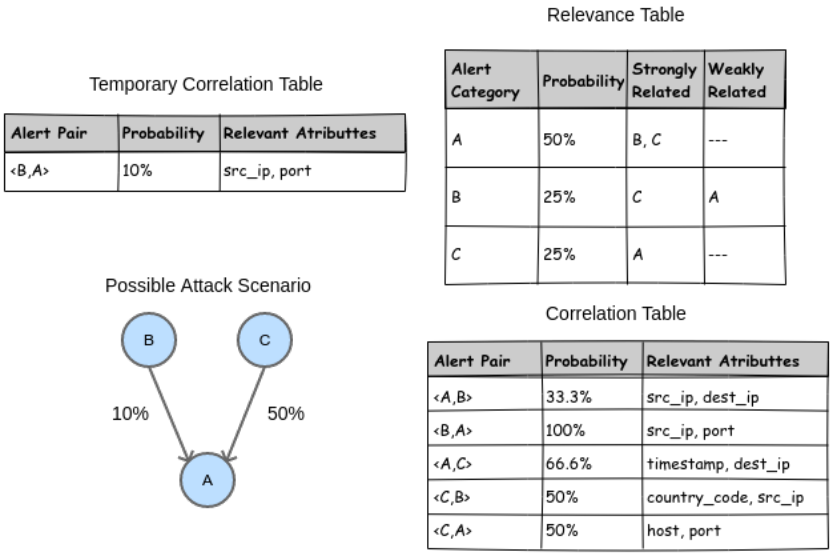


Figura 7.11: Criar Cenário de Ataque

Persistência dos modelos de dados estatísticos

Após os passos apresentados anteriormente obtemos modelos estatísticos que nos permitem entender:

- O estado normal da rede do cliente;
- A probabilidade de ocorrer cada tipo de alerta;
- A probabilidade de ocorrer um tipo de alerta depois de outro;
- As informações mais comuns quando ocorre um correlacionamento entre alertas;
- Quais os tipos de alertas que mais podem influenciar a ocorrência de outros;
- Variações do estado normal da rede do cliente;

Como os modelos estão em memória estes perdem-se cada vez que o Waldo é reiniciado, cancelando a criação de um modelo global do estado da rede.

Para mitigar este problema decidiu-se registar a RelevanceTable e a CorrelationTable na base de dados de investigações. Daqui surgiu o diagrama ER apresentado na figura 7.12.

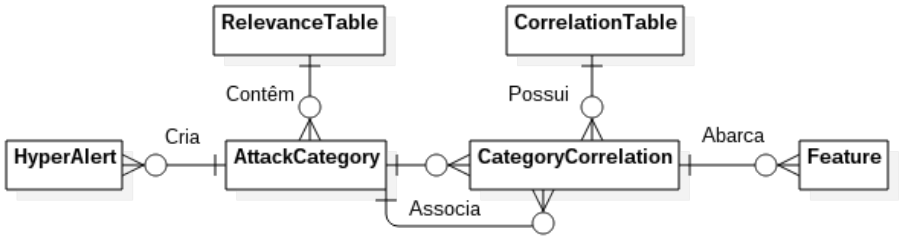


Figura 7.12: Statistical Models ER

Implementou-se então um mapeamento, utilizando postgresql para as queries necessárias e JDBC para conexão à base de dados a partir do ambiente de java, para os dados dos hiper alertas, das classes RelevanceEntry, CorrelationEntry, RelevanceTable, CorrelationTable e também dos atributos de modo a conseguir guardar os dados na base de dados a cada janela temporal, atualizar dados já existentes e carregar o estado da rede já guardado anteriormente.

Ao observar os resultados obtidos em 8.8 percebeu-se que o consumo de memória do Waldo com a implementação atual irá aumentar infinitamente devido à acumulação de hiper alertas. Realizando o registo destes na base de dados podemos remover quase todos os hiper alertas de memória. Iremos apenas manter os da janela temporal atual em memória para permitir o mais rápido cálculo probabilístico da janela temporal que é um cálculo bastante frequente e não acumula uma grande quantidade de hiper alertas. Contudo mesmo desta forma vamos realizar cálculos com muitos hiper alertas constantemente e vamos ter um grande volume de dados na base de dados. Com este problema em mente, percebeu-se que os dados que ficam guardados nas tabelas são objetos de RelevanceEntry e CorrelationEntry. Se associarmos um contador da frequência que vá acumulando a quantidade de hiper alertas que ocorreram do tipo da relevanceEntry conseguimos um cálculo muito mais barato para as probabilidades, além de que podemos a cada janela temporal destruir os hiper alertas da memória sem necessidade de os registar na base de dados, diminuindo em grande quantidade tanto o numero de cálculos necessários como a quantidade de dados que é necessário guardar e recarregar. O mesmo foi feito para as correlationEntry e para as features de cada uma destas resultando na estrutura apresentada na figura 7.13.

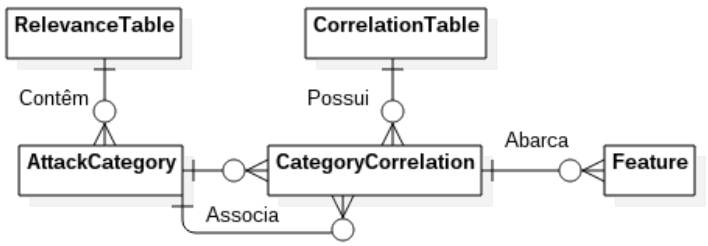


Figura 7.13: Statistical Models ER Without Hyper Alerts

7.4 Classificar Alertas

Cada correlacionador ao ser executado com novos alertas irá gerar modelos e análises. Estes modelos e análises variam de correlacionador para correlacionador, então para que possam ser analisados, compreendidos e classificados vamos precisar de um classificador propriamente adaptado para cada um deles.
Nesta secção são então apresentados os vários classificadores criados de modo a podermos usufruir da análise realizada por cada um dos correlacionadores.

7.4.1 Classificador de Resultados baseados em estatística

Este classificador irá analisar os resultados provenientes do correlacionador baseado em estatística sendo estes:

- Tabela de relevância com a possibilidade de ocorrer cada uma das categorias de alerta;
- Árvore representando um possível ataque com:
  - Nós dos vários alertas do ataque;
  - Probabilidades de causa entre os pares de nós correlacionados;
  - Atributos mais comuns nos nós do ataque.

Escolha do Classificador

Para conseguir realizar a classificação dos resultados anteriores necessitamos de analisar os dados e encontrar uma forma de perceber as diferenças entre as árvores de ataque e o que podem significar.
Deste modo deparamos-nos com os problemas:

- As categorias de alerta não são fixas e podem aparecer novas a qualquer altura;
- As features não são fixas e podem aparecer novas a qualquer altura;
- É complexo conseguir uma posição exata para cada cena de ataque;
- Existem poucos dados úteis disponíveis;
- É bastante complexa a criação de regras para estes tipos de dados;

Tendo estes problemas em mente foram analisadas várias abordagens e com base no artigo descrito em 3.2.6, percebeu-se que um sistema com base em CBR(Case Based Reasoning) poderia ser uma boa escolha visto que permite:

- Utilizar o conhecimento dos operadores para a própria classificação;
- Consulta posterior de modo a conseguir uma melhor compreensão dos dados;
- Cancelar a necessidade de uma posição global pois permite o cálculo da distância relativa entre cada um dos casos de ataque;
- Utilização com menos dados visto que pode ir guardando os novos casos à medida que recebe feedback dos operadores;
- Novas classes de categorias e features possíveis com uma implementação cuidada.

Implementação do Classificador

Agora já com a técnica base para ser utilizada pelo classificador decidida começou-se a sua implementação.
Inicialmente a prioridade foi representar os casos para poderem ser registados e comparados com os outros na base de dados, sendo para tal criada a classe AttackCase. Objetos desta classe são criados quando é detetado um comportamento estranho durante a análise do StatisticsCorrelator e com a receção de dados sobre a relevância de cada categoria associada ao possível ataque e à AttackScene correspondente, ficando com uma estrutura próxima do exemplo da figura 7.14.

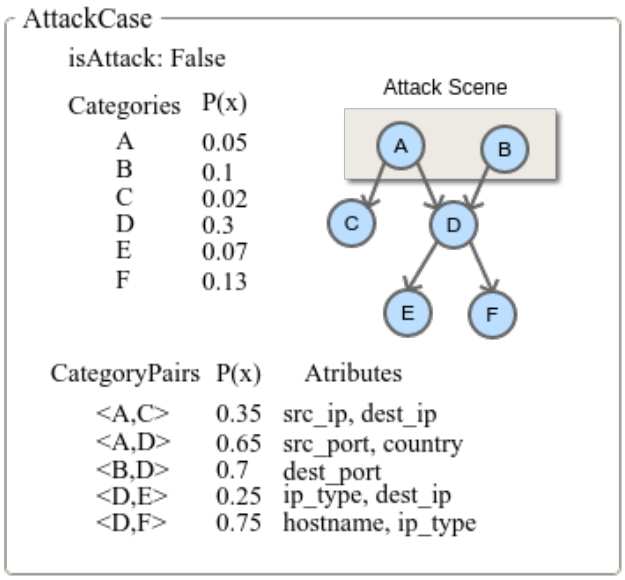


Figura 7.14: Exemplo AttackCase

Já com a estrutura base definida o próximo passo é perceber a semelhança entre dois casos. Para tal percebeu-se que o ponto mais importante para perceber a semelhança é a presença/ausência das categoria de ataque em ambos o ataques e também dos seus correlacionamentos. Com isto em mente, foi criado um cálculo de distância entre os casos de ataque como base numa combinação das seguintes técnicas:

- Cálculo de distância euclidiana[84] que permite o cálculo de distância entre quaisquer dois pontos, desde que tenham o mesmo numero de dimensões entre si;

$$euclidean\_dist(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{7.1}$$

- Cálculo de distância de hamming[85] pois esta permite medir a distância mínima de substituições a uma string binária para a transformar noutra. Como a comparação de distância é relativa entre dois pontos, podemos realizar esta comparação utilizando o valor binário resultante da existência ou não existência de categorias de alerta, pares categorias correlacionados e atributos entre os dois casos a comparar;

$$hamming\_dist(p,q) = \sum_{i=1}^{n} dist(p_i, q_i) \tag{7.2}$$

$$dist(p_i, q_i) = \begin{cases} 0 & \text{if } p_i = q_i \\ 1 & \text{if } p_i \neq q_i \end{cases} \tag{7.3}$$

- Normalização utilizando *Min-Max scaling* ou *Standardization*[86] para permitir a comparação das distâncias obtidas a partir dos diferentes tipos de cálculos com diferentes escalas;

$$Min\_Max(X) = \frac{X - Xmin}{Xmax - Xmin} \qquad Std(X) = \frac{X - \mu}{\sigma} \tag{7.4}$$

Para realizar este cálculo necessitamos de definir a importância de cada atributo do cenário de ataque, utilizando os pontos que se seguem como referência para decidir como a nossa distância será calculada:

- Visto que estamos a calcular uma distância relativa queremos entender quais os atributos que existem em cada caso. Um novo caso possuir novos dados não deve aumentar a distância entre os casos, podendo o antigo ser até um subset do que está em análise, contudo a falta de um atributo no novo caso dado que este existe nos casos já registados é sim uma distância entre estes;

- Para calcular a distância entre categorias, pares de categorias e atributos será tida como base a distância de hamming de modo a identificar se estas existem nos dois casos;

- Se as features ou pares de features existirem em ambos os casos vamos então calcular a distância entre as probabilidades utilizando a distância euclidiana;

- Ao utilizar a distância euclidiana ficamos com um valor numérico, contudo a escala entre este e o valor da distância de hamming é bastante díspar, para responder a este problema devemos aplicar algum tipo de normalização.

Começando pelo cálculo da distância entre diferentes probabilidades de ocorrer a mesma feature, foi utilizada a distância euclidiana recebendo p_i e q_i que são elementos do mesmo tipo com uma probabilidade associada a cada um. Visto que o cálculo foi utilizado em apenas uma dimensão, este pode ser simplificado para:

$$euclidean\_dist(p_i, q_i) = |P(p_i) - P(q_i)| \tag{7.5}$$

O resultado desta operação vai ser um valor que pode variar bastante em relação aos valores obtidos com a distância de hamming, para resolver este problema vamos normalizar o resultado. Dado que estamos perante um

ambiente que avalia as mudanças de probabilidades ao longo do tempo, escolheu-se, em vez de utilizar as abordagens de normalização comuns, utilizar a variação da probabilidade em relação à probabilidade do caso já existente para medir a distância entre estas.

$$\Delta P(p_i, q_i) = \frac{|P(p_i) - P(q_i)|}{P(p_i)} \tag{7.6}$$

Se houver uma grande variação na probabilidade o valor da distância resultante de $\Delta P(p_i, q_i)$ pode ultrapassar o valor máximo da distância de hamming que vai representar se algo existe ou não com os valores 1 e 0 respetivamente. Para manter uma relação entre as escalas dos diferentes tipos de cálculos deve ser tido em conta que mesmo que exista uma grande diferença entre as probabilidades,o valor de distância deve ser sempre inferior ou igual ao valor no caso de inexistência. Deste modo foi colocado um limite superior de distância 1 para quando a probabilidade tiver uma variação superior ou igual a 100%:

$$\Delta P\_lim(p_i, q_i) = \begin{cases} \Delta P(p_i, q_i) & \text{if } \Delta P(p_i, q_i) < 1 \\ 1 & \text{if } \Delta P(p_i, q_i) >= 1 \end{cases} \tag{7.7}$$

Acima estava a ser considerada a diferença das probabilidades de categorias existentes, contudo devemos também considerar que essa distância apenas pode ser calculada caso exista a mesma categoria no antigo AttackCase e no novo AttackCase. Para representar esta existência foi utilizada a distância de Hamming, devolvendo 0 no caso da categoria existir em ambos e 1 se não existir no novo AttackCase. Dado que associado à existência da categoria temos de seguida associada a distância da probabilidade desta ocorrer, devemos juntar ao valor da distância de hamming a distância obtida a partir da variação da probabilidade.

$$element\_dist(p_i, q_i) = \begin{cases} \Delta P\_lim(p_i, q_i) & \text{if } p_i = q_i \\ 2 & \text{if } p_i \neq q_i \end{cases} \tag{7.8}$$

Com este passo conseguimos uma generalização do cálculo da distância que nos permite comparar qualquer elemento que tenha uma probabilidade associada, sendo já suficiente para calcular isoladamente a distância entre categorias, pares de categorias e atributos de AttackCases. No entanto a distância dos atributos deve ser associada ao valor dos pares de categorias visto que estes existem juntos e não devem ser separados, de modo que surgiu o seguinte cálculo para calcular a distância entre os atributos de diferentes pares de categorias, sendo p_i e q_i pares de categorias e $m = len(p_i.features)$.

$$attribute\_dist(p_i, q_i) = \frac{\sum_{j=1}^{m} element\_dist(p_i.features_j, q_i.features_j)}{m * 2} \tag{7.9}$$

Agora já com o cálculo de atributos criado, para o cálculo da distância completa entre pares de categorias basta adicionar a este a distância entre os próprios pares de categorias e as suas probabilidades que pode ser feito com a função da distância entre probabilidades.

$$pair\_dist(p_i, q_i) = \begin{cases} \Delta P\_lim(p_i, q_i) + attribute\_dist(p_i, q_i) & \text{if } p_i = q_i \\ 2 & \text{if } p_i \neq q_i \end{cases} \tag{7.10}$$

Após todas estas considerações construiu-se um cálculo final de distância entre casos de ataque. Tendo em conta que α representa um AttackCase já guardado no sistema, sendo new_α um novo AttackCase, $n = len(\alpha.categories)$ e $m = len(\alpha.pairs)$, este cálculo pode-se traduzir na expressão:

$$case\_dist(\alpha, new\_\alpha) = \sum_{i=1}^{n} element\_dist(\alpha.categories_i, new\_\alpha.categories_i) + \sum_{i=1}^{m} pair\_dist(\alpha.pairs_i, new\_\alpha.pairs_i) + \Delta P\_lim(\alpha.total\_nodes, new\_\alpha.total\_nodes) \tag{7.11}$$

Persistência dos resultados

Agora já com capacidade de comparar AttackCases o próximo passo foi a construção de um modelo de dados para estes casos de modo a podermos classificar novos casos de ataque, apresentar aos operadores no Waldo Web e classificar estes casos com o feedback do operador. O modelo criado pode ser observado na figura 7.15.

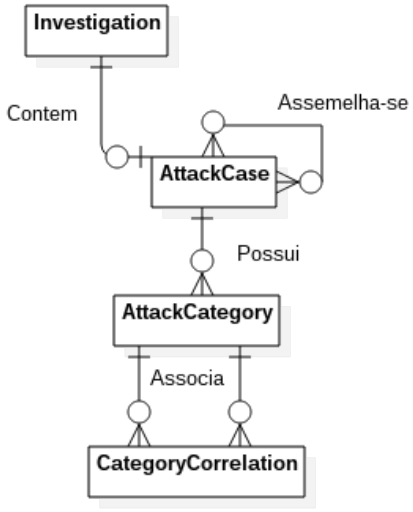


Figura 7.15: Diagrama ER do modelo de dados associado ao AttackCase

- **Investigation** - Responsável por representar uma investigação e por a apresentar ao operador. Vai estar ligado a todos os classificadores criados;
- **AttackCase** - Representa o caso de ataque construído no classificador estatístico, estando ligado aos casos de ataque mais semelhantes a este, possuindo também as categorias de ataque e os pares de ataque que estavam associados ao mesmo;
- **AttackCategory** - É uma categoria de ataque associada a um caso de ataque com a sua descrição e a probabilidade de ocorrer;
- **CategoryCorrelation** - Associa um par de categorias a um caso de ataque, apresentando a probabilidade de a segunda categoria acontecer após a primeira e registando os atributos mais comuns quando esta ligação acontece.

Agora já com o conhecimento dos dados da nossa classe AttackCase e com o ER para demonstrar os relacionamentos, implementou-se um mapeamento dos dados da classe, utilizando postgresql para as queries necessárias e JDBC para conexão à base de dados a partir do ambiente de java, para a base de dados de investigações. Esta implementação permite guardar ataques quando estes forem detetados e carregar ataques quando o Waldo for ligado e estiver a inicializar o classificador com os AttackCases já existentes.
Contudo nem todos os ataques gerados devem ser guardados. Quando um

novo possível ataque for muito próximo a um já registado e este tiver uma grande probabilidade de não ser uma ameaça, pode ser ignorado. Esta medida permite limitar a existência de casos quase semelhantes que sem grande benefício iriam aumentar bastante o custo do cálculo dos novos casos.

7.5 Apresentar Investigações

Nesta secção serão descritas as tarefas para implementar a funcionalidade de apresentar investigações aos operadores, a construção da base de dados de investigações, a criação da lógica responsável por buscar as investigações e responder aos pedidos dos operadores e as vistas web para visualização das investigações.

7.5.1 Investigations Database

Para a base de dados do Waldo Web será aproveitado o modelo ORM utilizado pela framework Django que permite a criação da base de dados e da lógica associada à persistência dos dados de modo transparente a partir das classes e objetos em python. Tendo em conta os modelos ER apresentados em cada correlacionador e classificador, foi criado o diagrama de classes apresentado na figura 7.16 que irá ser utilizado para construir a base de dados do Waldo Web.

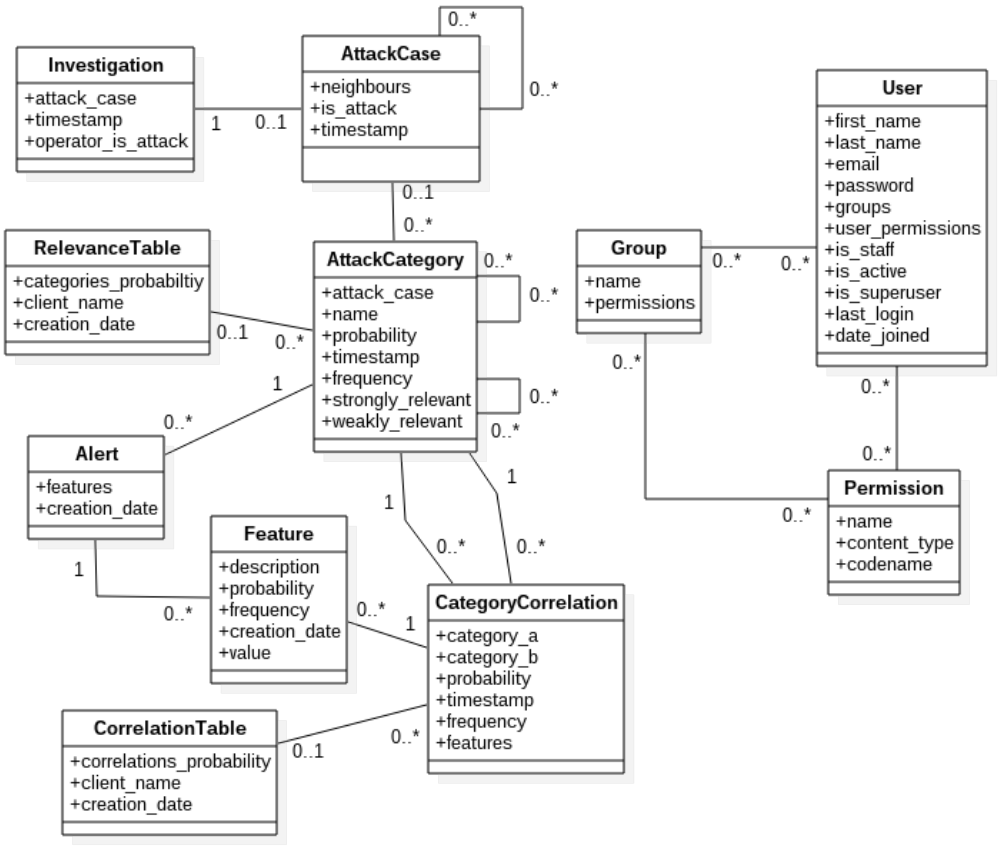


Figura 7.16: Diagrama de classes do Waldo Web

As classes de User, Group e Permission são criadas automaticamente pelo motor ORM do django ao incluir o módulo de autenticação[87]. Graças a este módulo poupamos tempo na implementação de funcionalidades destas classes. Ganhamos também estabilidade, visto que este módulo está bastante testado pela sua grande utilização por parte da comunidade e ganhamos pela simplicidade de atualização caso hajam modificações no próprio django, visto que este é um dos módulos do core da framework e irá evoluir com a mesma.
Quanto às outras classes, estas apresentam o necessário para representar as investigações e os resultados de cada um dos classificadores demonstrados na secção 7.4.

7.5.2 Investigations Presentation Business Logic Module

A camada de lógica de negócio do Waldo Web é responsável por apresentar as páginas pedidas, fornecer ao utilizador informações obtidas a partir da

base de dados de investigações, gerir os utilizadores e as suas permissões. Para entender os pedidos de utilizador começamos pelo mapeamento de URLs. Este mapeamento é realizado com auxilio do *URL dispatcher*[88] do django que permite de forma simples mapear expressões regulares para as views do django. Este procura permitir a construção de URIs que não mudem ao longo do tempo, como apresentado por *Tim Berners-Lee* em [89]. Criaram-se então URLs para as principais informações a visualizar e ações a realizar:

- Página de Login;
- Ação de Login;
- Ação de Logout;
- Página de Dashboard de Investigações;
- Página de Categoria de Alerta;
- Página de Investigação;
- Ação de buscar dados de investigação;
- Ação de buscar dados do cliente;

A cada um destes está associada uma view que:

- No caso das páginas faz o render de um template de django e envia a representação html para o browser do utilizador;
- Na ação de Login recebe um POST que caso tenha dados corretos faz login e reencaminha para a página de Dashboard de Investigações, senão devolve uma mensagem de erro em formato JSON;
- Na ação de Logout, limitada apenas para utilizadores que estejam logados, faz logout do utilizador e redireciona-o para a Página de Login;
- A ação de buscar dados de investigação, apenas disponível para utilizadores logados, vai receber um POST com o id referente à investigação pretendida, vai buscar o objeto correspondente, vai realizar a sua conversão para o formato JSON e vai responder com essa informação para o cliente;
- A ação de buscar dados do cliente, apenas disponível para utilizadores logados, vai receber um POST com o tipo de dados pretendidos e o cliente associado, respondendo de seguida com os dados pedidos no formato JSON se todos os parâmetros forem válidos, ou com uma resposta de erro caso algum não o seja. Os dados possíveis do cliente que podem ser pedidos são:

– A Lista de Investigações;
– A Tabela de Relevância Global;
– A Tabela de Correlacionamente Global.

Para esta ação foi utilizado o padrão estratégia[90], criando-se uma estratégia para buscar os dados do cliente com três implementações diferentes. Através desta abordagem podemos decidir em run-time quais os dados de cliente a buscar e a forma como estes devem ser processados, enquanto marcamos de forma simples que todas estas implementações mesmo apresentando comportamentos diferentes tem o mesmo objetivo.

De modo a simplificar as conversões dos vários tipos de dados pretendidos para formato JSON foi criado o método get_json() em cada um dos modelos de dados, de modo a que todos tenham uma representação direta em formato JSON.
Podemos observar então um panorama geral do funcionamento da camada de lógica de negócio do Waldo na figura 7.17.

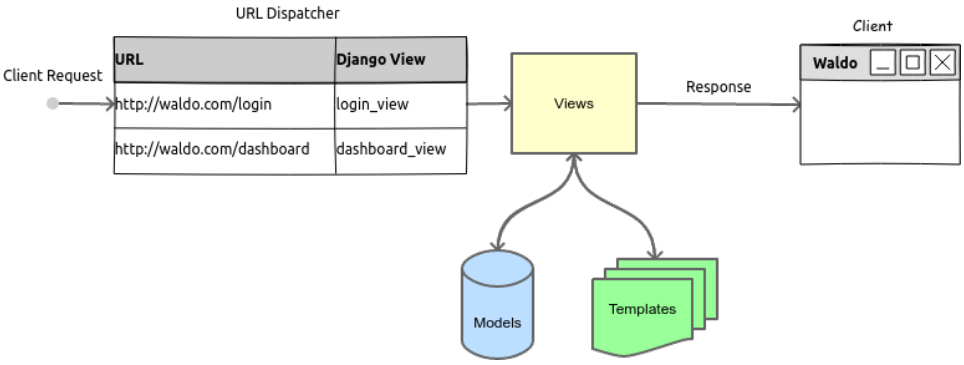


Figura 7.17: Waldo Web Business Layer

7.5.3 Waldo UI

A interface de utilizador consiste nos templates de django construidos. Estes templates utilizam a linguagem de templates do Django[91], em que um template:

- Pode ser de qualquer tipo de ficheiro de texto (HTML, XML, CSV, etc);
- Contém variáveis que vão ser trocadas por valores quando o template for renderizado pelas views;
- Contém tags que controlam o fluxo do template;

- Contém filtros que podem ser aplicados às variáveis.

Para facilitar a perceção da estrutura destes, um pequeno exemplo pode ser visto na figura 7.18. Este exemplifica a criação de uma lista em html com o conteúdo do campo type para cada alert na alert_list.

```
<ul>
{% for alert in alert_list %}
  <li>{{ alert.type }}</li>
{% endfor %}
</ul>
```

Figura 7.18: Django template language example

Consoante as páginas especificadas na secção 7.5.2, vamos ter como templates:

- O template de Login;
- O template de Dashboard de Investigações;
- O template de Categoria de Alerta;
- O template de Investigação.

Login

Para permitir ao utilizador fazer login foi criado o template de Login. Este apresenta uma página agradável onde se pode inserir o username e password para aceder às investigações da plataforma. A partir deste template é gerada a página apresentada na figura 7.19.

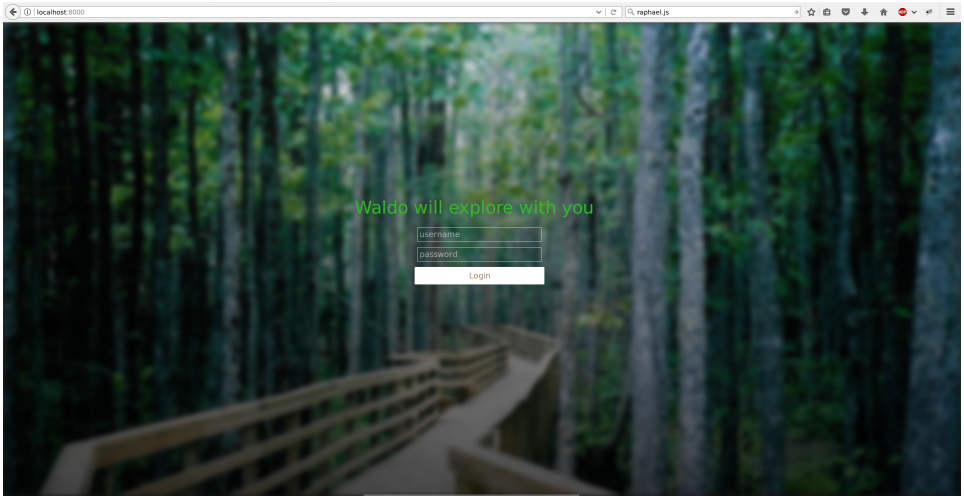


Figura 7.19: Login page

Ao submeter os dados são enviados para a ação de login. Esta no caso de receber dados corretos redireciona para a dashboard de investigações e no caso de receber dados errados apresenta uma mensagem de erro como se pode ver na figura 7.20.

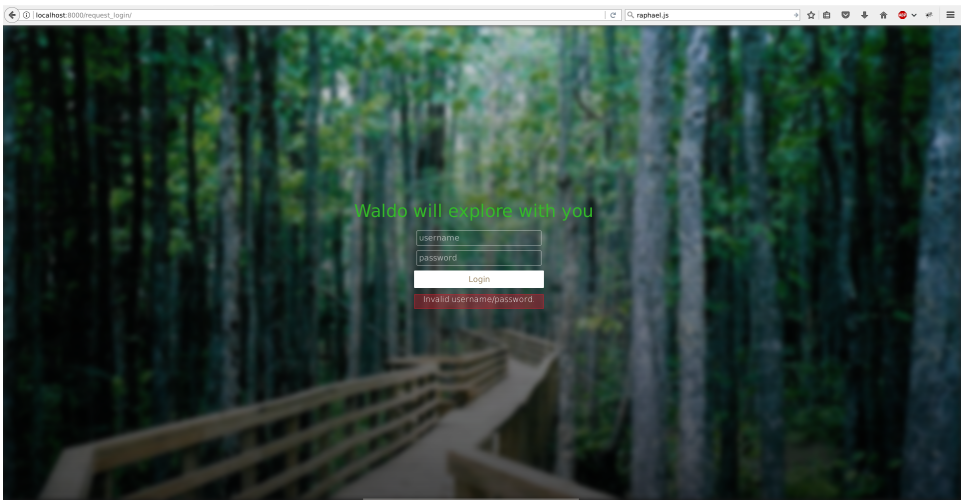


Figura 7.20: Login page error

Dashboard de Investigações

A dashboard de investigações tem 5 áreas principais. Estas foram construidas como objetos olhando para o funcionamento dos componentes de React[92] com o intuito de facilitar a criação, manutenção e utilização destes[93]. Para tal, foram definidas as seguintes regras:

- **Cada área recebe um elemento jquery como contentor** - Este será a área que o objeto pode controlar de modo a conseguir desenhar-se e estruturar o seu conteúdo;
- **Cada área possui o método update** - Atualiza os dados do objeto, normalmente pedindo dados ao servidor;
- **Cada área possui o método render** - Limpa o conteúdo do seu contentor e redesenha o objeto com os seus dados atuais.

As 5 áreas são:

- **Lista de Clientes** - Apresenta os vários clientes existentes, e quando um destes é clicado todas as outras áreas se atualizam para os dados do cliente selecionado. Esta área pode ser visualizada na figura 7.21;

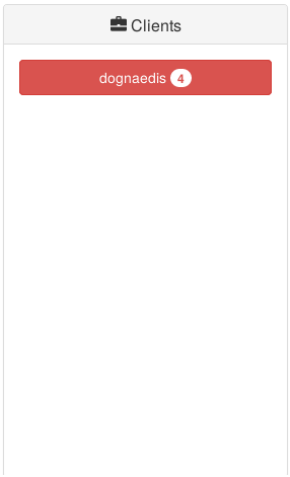


Figura 7.21: Investigations Clients List

- **Lista de Investigações** - Apresenta a lista de investigações registadas pelo sistema Waldo Investigations e que ainda não receberam feedback. Quando selecionada uma destas, é atualizada a área da investigação atual. Pode-se ver esta lista na figura 7.22;

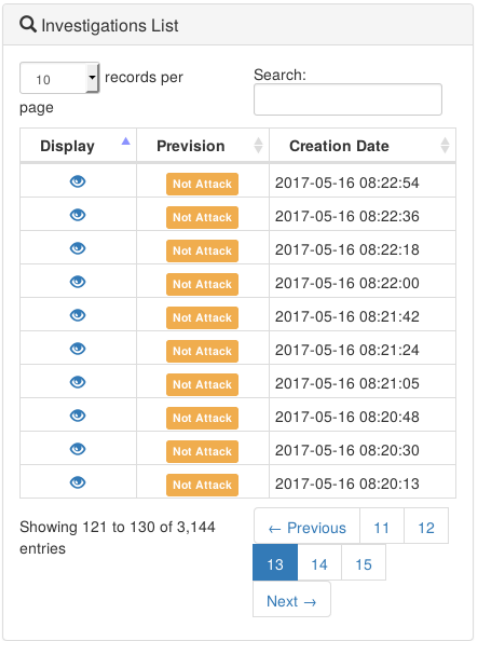


Figura 7.22: Investigations List

- **Investigação Atual** - Possui uma descrição da causa que originou a investigação e um grafo que representa a investigação em si. Para obter a melhor representação possível de modo a facilitar a compreensão

do problema apresentado pela investigação foram construidos vários protótipos e testes com diferentes ferramentas para grafos. As ferramentas testadas foram:

- **arbor.js** - Permite a representação de ligações da investigação e foi simples de utilizar. Não permite uma representação de hierarquia, o que vai limitar a capacidade de análise da investigação pois pretende-se entender as relações de causalidade entre os vários tipos de alertas. Pode-se visualizar uma investigação com esta ferramenta na figura 7.23;

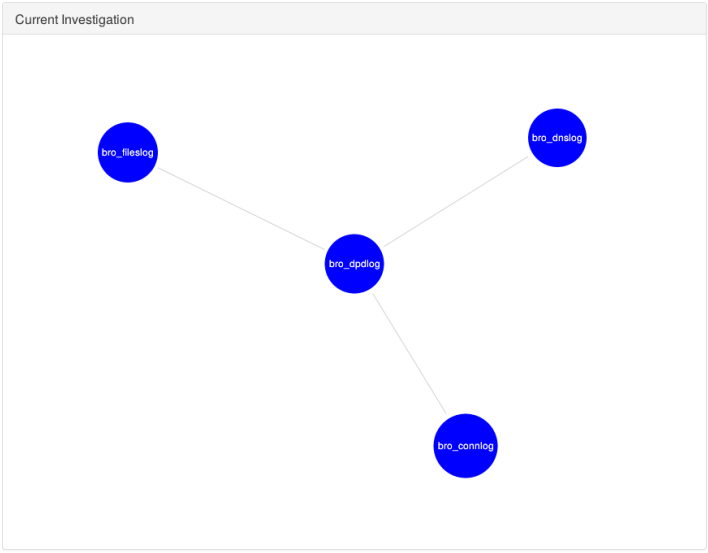


Figura 7.23: Investigations with arbor.js

- **treant.js** - Permite a representação hierárquica dos alertas, contudo exigiu uma maior complexidade na construção de alertas. Não permite colocar texto nas ligações, o que seria bastante útil para representar as probabilidades de correlacionamento entre alertas. Esta espera a construção de árvores que se vão sempre sempre ramificando, então não permite que dois ou mais alertas no nível acima se liguem simultaneamente ao mesmo alerta no nível abaixo. Podemos ver como é a visualização da investigação com esta ferramenta na figura 7.24;

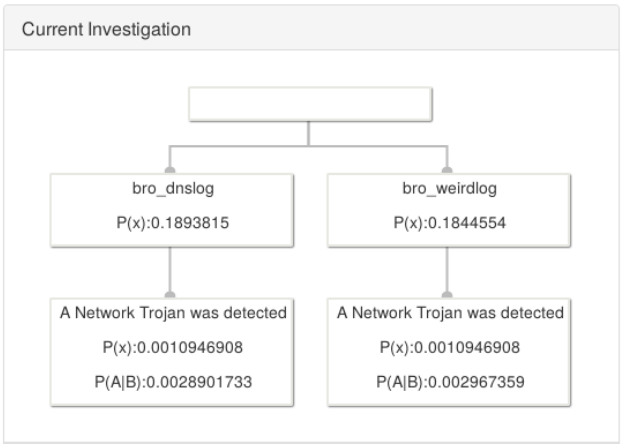


Figura 7.24: Investigations with treant.js

– **vis.js** - Permite a construção de grafos com hierarquia, apresentar texto nas ligações entre nós, despoletar eventos como left_click, right_click, double_click, hover, entre outros. Permite destacar cada nó e ligação individualmente permitindo a identificação fácil da causa da investigação ter sido despoletada. Deste modo foi a ferramenta selecionada pois consegue colmatar todas as faltas existentes para a representação de investigações do arbor.js e do treant.js enquanto ainda oferece outras funcionalidades, como podemos ver na figura 7.25.

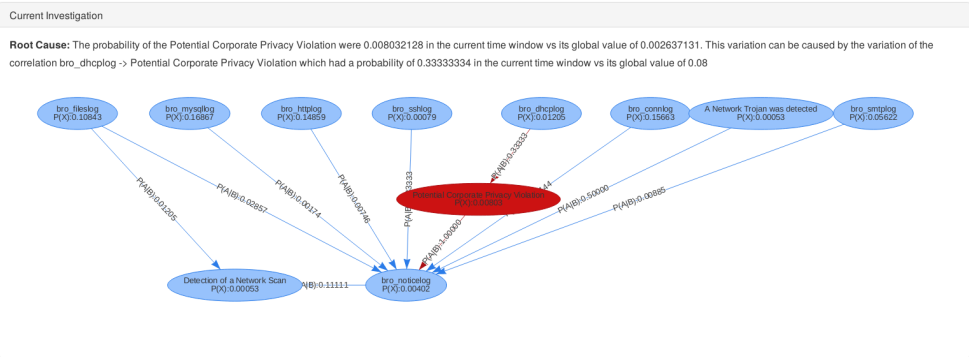


Figura 7.25: Investigations with vis.js

- **Tabela global de Relevância** - Apresenta a lista com as probabilidades globais de cada tipo de alerta que tenha sido recebido pelo Waldo Investigações para o cliente atualmente selecionado, como se pode ver figura 7.26.

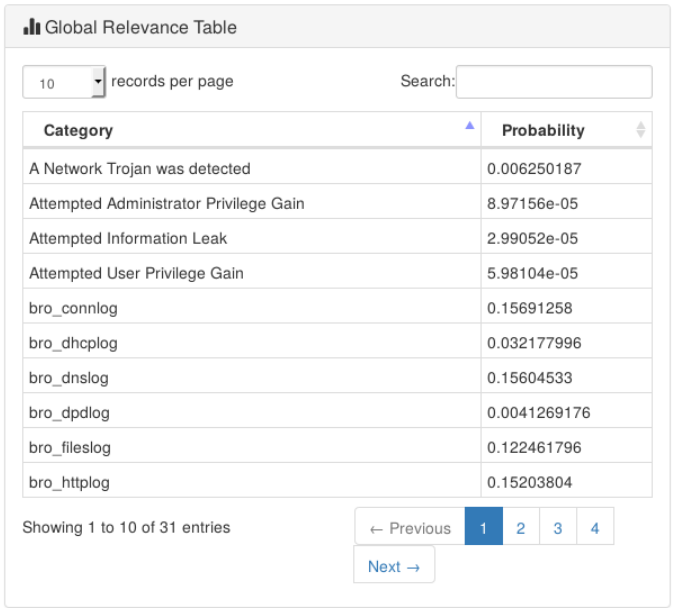


Figura 7.26: Tabela de Relevância

- **Tabela global de correlações** - Apresenta as probabilidades globais de correlacionamentos entre categorias de alertas do cliente atualmente selecionado, como se pode ver na figura 7.27;

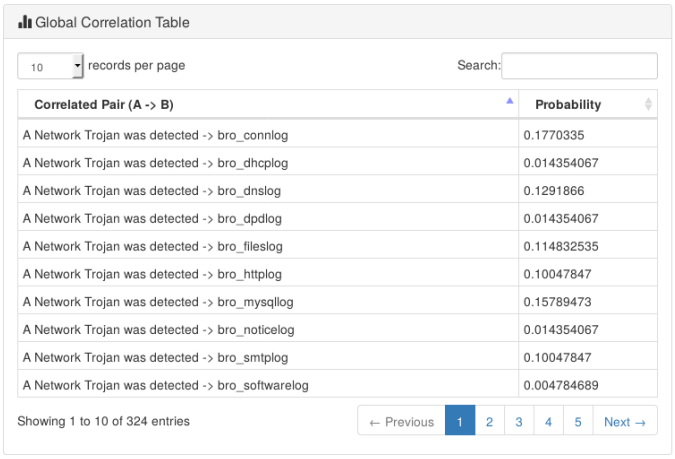


Figura 7.27: Tabela de correlações

Agregando os objetos das 5 áreas obtemos uma página em que cada parte consegue atualizar-se isoladamente e de forma assíncrona. Toda a página torna-se simples de estruturar pois basta apenas mudar os contentores que são passados para os componentes, que estes se encarregam de gerir o contentor recebido e desenhar-se nesse mesmo espaço.

Obteve-se como resultado da Dashboard de investigações a página apresentada na figura 7.28.

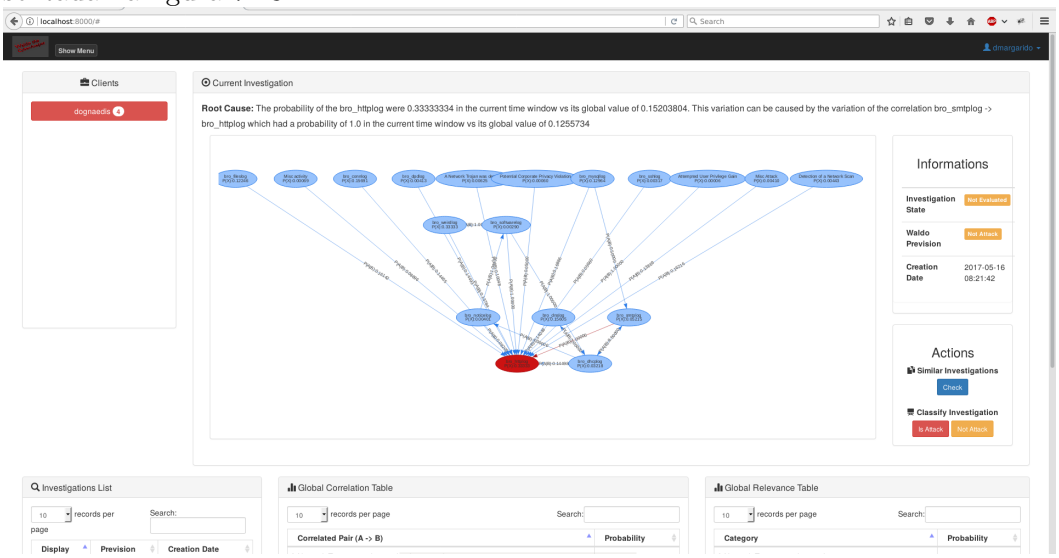


Figura 7.28: Dashboard de investigações

Categoria de Alerta

Ao clicar nos nós de categorias de alerta apresentados na área de investigação é aberta uma nova página em relação a esta mesma categoria. Aqui podem ser vistos os dados relacionados com esta categoria de alerta durante a janela temporal existente aquando da criação da investigação. Esta página apresenta:

- O nome da categoria de alerta, a probabilidade desta ocorrer na janela temporal existente no momento em que a investigação foi criada e o momento da criação, como apresentado na figura 7.29;

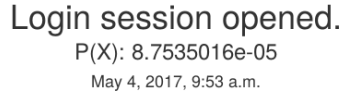


Figura 7.29: Descrição de Categoria

- As correlações da categoria de alerta selecionada durante a investigação atual e as suas probabilidades. Estas podem ajudar a identificar os tipos de logs que podem ter sido gerados após a atividade suspeita, permitindo uma rastreabilidade do impacto e da forma de atuar, ou por exemplo, a correlação entre várias atividades suspeitas, como se pode ver na figura 7.30;

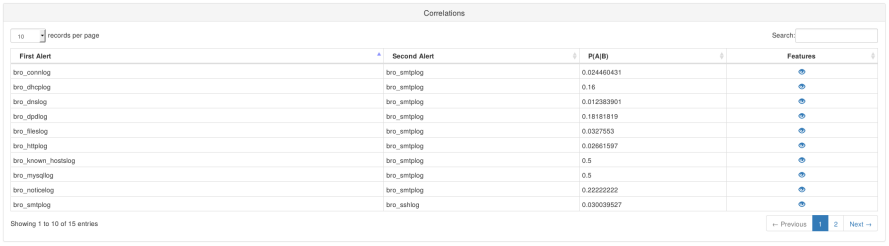


Figura 7.30: Correlacionamento de Categorias

- Ao selecionar a correlação são apresentados os atributos mais comuns que existiram naquela correlação, como se pode ver na figura 7.31;

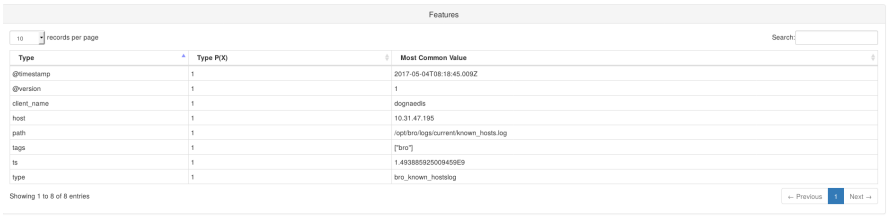


Figura 7.31: Atributos de correlacionamento de categorias

- Uma lista com os alertas da categoria atual e todos os seus atributos, recebidos dos IDS que estão associados à investigação e sua janela temporal. Estes são uma lista agregada dos alertas despoletados pelos IDS associados à categoria atual e associados à investigação de modo que facilita a pesquisa e investigação, como se pode ver na figura 7.32;

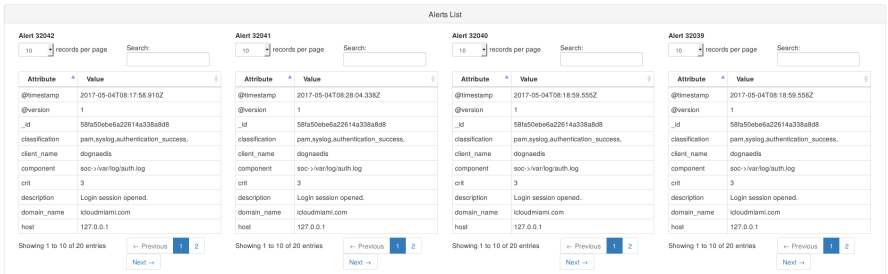


Figura 7.32: Lista de alertas

- Um componente de pesquisa de alertas que permite filtrar os alertas por qualquer texto inserido. Este componente, apresentado na figura 7.33, vai realizar a comparação tanto com atributos como com os seus valores. Deste modo quando existir uma grande quantidade de alertas e se encontrar algo suspeito, podemos limitar o que é apresentado apenas ao que interessa ver e não temos de pesquisar um alerta de cada vez para verificar se está relacionado com o que é pretendido.

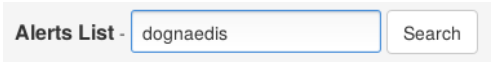


Figura 7.33: Pesquisa de Alertas da Categoria

Investigação

De modo a facilitar a análise da investigação atual e também para a poder consultar mais tarde, foi criada uma página individual para cada investigação. Nesta vai ser apresentada a investigação selecionada e as investigações passadas que forem mais parecidas com a mesma de modo a permitir uma análise comparativa para se entender mais facilmente o porquê da investigação e qual o problema a classificar. Associada a cada uma das investigações passadas é apresentada a distância, esta é a métrica de semelhança entre investigações calculada pelo classificador estatístico do Waldo Investigações. A página pode ser visualizada nas figuras 7.34 e 7.35.

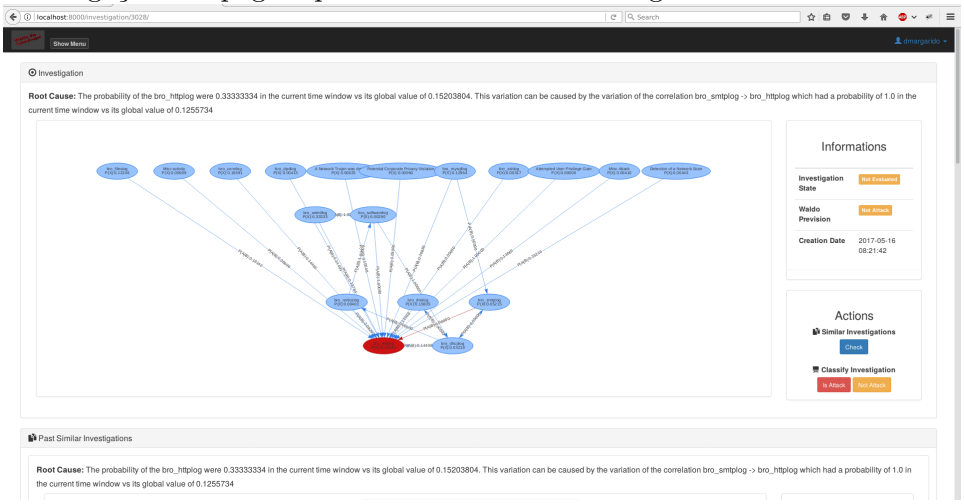


Figura 7.34: Investigation Page 1

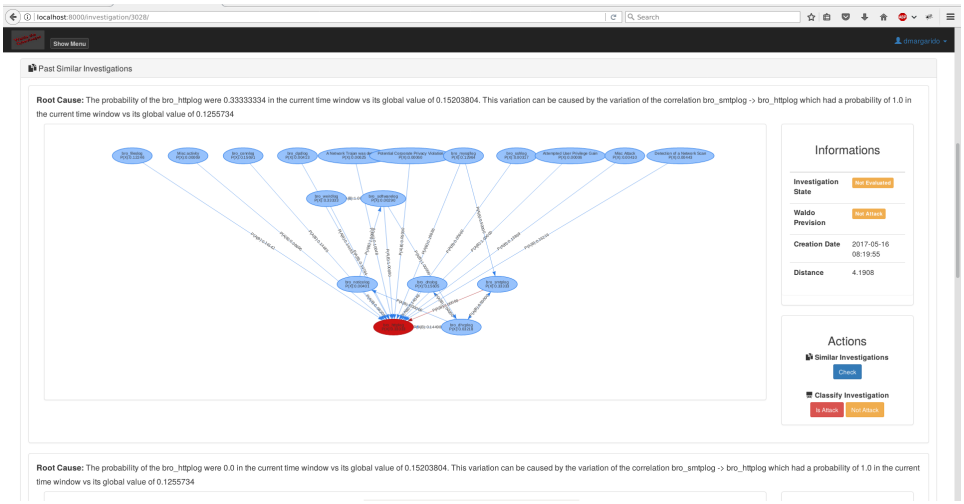


Figura 7.35: Investigation Page 2

7.6 Aumentar Precisão de Classificação

De modo a garantir o aumento da qualidade e avaliação das investigações do Waldo ao longo do tempo, este necessita de ir obtendo feedback sobre cada investigação. Utilizando o sistema baseado em casos que foi construido, este vai aprender com os novos casos avaliados pelos operadores como se pode confirmar na secção 8.2.1. Para conseguir fornecer essa avaliação foi criado:

- Componente de interface associado a cada investigação não avaliada que permita a submissão do feedback. Esta pode ser observada na figura 7.36;

Figura 7.36: Classify Investigation

- Uma implementação de um publisher utilizando o cliente pika de rabbitMQ em python para poder enviar o feedback para a queue do cliente;

- Url e view para registar estes dados na base de dados e reportar para o Waldo Investigações. Esta view inicialmente valida se o utilizador está logado, se é um utilizador conhecido, se a investigação passada existe e se esta ainda não foi avaliada. Se passar nestas validações, vai então definir a investigação como avaliada e atribuir-lhe a classificação dada pelo operador, utilizando por fim o publisher construido para enviar a

mensagem do feedback com o formato JSON para a queue do cliente associado à investigação;

- Processamento da mensagem de feedback na queue de cada cliente, priorizando o feedback durante o Alerts Gathering State de modo a que seja rapidamente interpretado o feedback para que possa ser utilizado já nas próximas investigações;
- Um algoritmo de pesquisa para encontrar o AttackCase em memória associado ao feedback recebido e atualizar a avaliação do mesmo.

7.7 Automatizar Tarefas

Para a automatização de tarefas foi implementado o registo de links associados às categorias de alertas.
A funcionalidade de adicionar um novo link e remover link na página da categoria, permite simplificar a próxima análise a alertas do mesmo tipo. Ao utilizar os links iremos encontrar diretamente o material utilizado para resolver o problema na ultima vez que foi analisado, diminuindo o tempo de pesquisa para encontrar o material correto, como podemos confirmar na secção 8.2.3.
Pode ser visualizado o acesso aos links da categoria na figura 7.37.

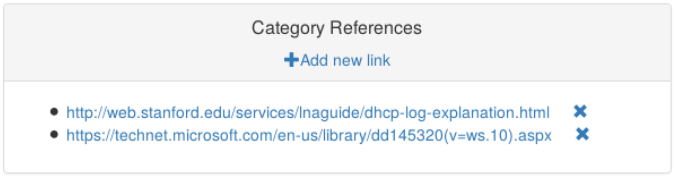


Figura 7.37: Links de Categoria

7.8 Conclusão

Neste capitulo foram apresentadas as dificuldades de cada uma das etapas do desenvolvimento e os passos tomados para as superar. Conseguiu-se a construção de um sistema com todos os pontos principais pretendidos neste projeto:

- Foi obtido um sistema capaz de deduzir possíveis ataques de forma autónoma, considerando apenas os dados recebidos pelos IDS;
- O sistema é capaz de criar um modelo do comportamento das redes dos clientes e a partir destes modelos detetar comportamentos suspeitos;
- O Waldo consegue utilizar e amplificar o conhecimento recebido dos operadores;

- Foi construido um sistema simples de utilizar e com uma interface agradável.

Capítulo 8

Verificação e Validação

Neste capitulo vai ser realizada a verificação e validação do sistema[94].

- Validação - Garante que um produto, serviço ou sistema corresponde às necessidades do cliente e outros stakeholders identificados. Regularmente envolve aceitação e adequação com clientes externos. - "*Are you building the right thing?*"[95]
- Verificação - Avalia se um produto, serviço ou sistema cumpre com uma regulação, requisito, especificação, condição imposta. Normalmente é um processo interno. - "*Are you building it right?*"[95]

Esta análise está partida em várias secções de teste(ST), possuindo cada uma destas a seguinte estrutura:

- **Tipo de teste** - Tipo de teste a usar consoante os tipos apresentados e detalhados no apêndice C;
- **Objetivos** - Quais os objetivos da abordagem de teste escolhida;
- **Modelos** - Cada técnica de teste necessita de algum tipo de modelos para análise;
- **Plano de Teste** - Definir o domínio a testar, descrever e justificar a estratégia para testar considerando o nível de coverage, ferramentas disponíveis, entre outros;
- **Casos de Teste** - Explicar como os casos de teste são definidos;
- **Execução** - Descrever a forma como os testes foram construidos e de que modo foi efetuada a validação dos resultados, incluindo uma explicação das ferramentas utilizadas;
- **Resultados de Teste** - Explicar o resultados de teste, apresentando coverage, bugs encontrados, entre outros;

- **Análise** - Analisar todo o processo de teste, resultados atingidos, dificuldades, entre outros.

8.1 Testes dos Requisitos Funcionais

8.1.1 ST01 - Teste aos componentes de Waldo Investigações

- **Tipo de teste** - Teste de Coverage;
- **Objetivos** - O Waldo Investigações para funcionar precisa de toda uma interação e integração do sistema sem falhar. Para garantir a qualidade das funcionalidades implementadas, simular o comportamento dos elementos mesmo sem dados, explorar de forma simples e rápida casos especiais, garantir que após modificações em código antigo, este continua funcional e para garantir que os componentes construidos interagem sem problema entre eles, procurou-se a implementação de testes para a maioria dos componentes;
- **Modelos** - Linhas executáveis por package 8.1:

Tabela 8.1: Linhas executáveis por package

Package	**Executable Lines**
InvestigationTest	791
WaldoInvestigation	67
WaldoInvestigation.Alerts	141
WaldoInvestigation.AlertsBroker	105
WaldoInvestigation.StateMachine	164
WaldoInvestigation.StateMachine.ClassificationModule	25
WaldoInvestigation.StateMachine.ClassificationModule.StatisticsClassifier	246
WaldoInvestigation.StateMachine.CorrelationModule	18
WaldoInvestigation.StateMachine.CorrelationModule.StatisticsCorrelator	729
WaldoInvestigation.StateMachine.GatheringModule	57
Total	2343

- **Plano de Teste**:
  - Coverage:

    Um objetivo mínimo de 80% de coverage foi definido para o sistema.
  - Domínio de Teste:

    O domínio de teste são todas as classes dentro do módulo Waldo Investigações e os testes associados ao mesmo.

- **Casos de Teste** - Devido à sua extensão são apenas aqui apresentados algumas amostras dos testes existentes:
  - Caso de teste 1 - Testar o parser de JSON;
  - Caso de teste 2 - Testar a construção de alertas a partir dos alertas JSON recebidos;
  - Caso de teste 3 - Testar envio de alertas para as queues dos clientes;
  - Caso de teste 4 - Testar as transições entre estados da máquina de estados;
  - Caso de teste 5 - Testar o registo de investigações na base de dados;
  - Caso de teste 6 - Testar capacidade do correlacionador estatístico de detetar variações suspeitas nas probabilidades.
- **Execução**:
  - Ferramentas Utilizada:

    JUnit 4;

    Intellij IDEA.
  - Casos de Teste:

    Caso de teste 1 - O parser de json parsou duas a três amostras de alertas do Bro, Snort e Ossec e não gerou nenhuma excepção;

    Caso de teste 2 - O consumidor da queue externa recebeu um alerta de cada tipo em JSON e utilizando a Alert factory construiu os alertas com as classes corretas;

    Caso de teste 3 - O sistema recebeu um alerta de cada tipo na queue externa em JSON e enviou os objetos construidos para o cliente correto, confirmando no consumidor do cliente que estes foram recebidos;

    Caso de teste 4 - A máquina de estados faz uma iteração em cada um dos estados, exercitando a transição entre os vários estados e confirmando o fluxo suposto;

    Caso de teste 5 - Foi gerado um comportamento suspeito que por sua vez despoletou uma análise no classificador estatístico. Este identificou o comportamento como sendo suspeito, o que gera uma investigação e realiza o seu registo na base de dados do Waldo Web;

    Caso de teste 6 - Foram produzidos alertas de modo a construir um estado inicial probabilístico para o correlacionador estatístico e de seguida foram gerados alertas em maior quantidade de modo a criar uma modificação brusca das probabilidades.

com esta maior alteração de probabilidades foi identificado um comportamento suspeito pelo correlacionador estatístico, confirmando que os valores de probabilidade coincidiram com o esperado.

- **Resultados de Teste**:

  Todos os testes passaram;

  Foi obtida uma percentagem de line coverage de 83.8%, sendo esta superior ao pretendido;

  Coverage por package na figura 8.1.

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	87,8% (43/ 49)	87,6% (269/ 307)	83,8% (1964/ 2343)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
InvestigationTest	100% (12/ 12)	95,9% (71/ 74)	91,3% (722/ 791)
WaldoInvestigation	25% (1/ 4)	11,1% (1/ 9)	10,4% (7/ 67)
WaldoInvestigation.Alerts	85,7% (6/ 7)	86,4% (19/ 22)	89,4% (126/ 141)
WaldoInvestigation.AlertsBroker	100% (4/ 4)	100% (15/ 15)	81% (85/ 105)
WaldoInvestigation.StateMachine	66,7% (4/ 6)	73% (27/ 37)	65,9% (108/ 164)
WaldoInvestigation.StateMachine.ClassificationModule	100% (2/ 2)	100% (5/ 5)	88% (22/ 25)
WaldoInvestigation.StateMachine.ClassificationModule.StatisticsClassifier	100% (2/ 2)	90,3% (28/ 31)	87,8% (216/ 246)
WaldoInvestigation.StateMachine.CorrelationModule	100% (2/ 2)	100% (4/ 4)	100% (18/ 18)
WaldoInvestigation.StateMachine.CorrelationModule.StatisticsCorrelator	100% (8/ 8)	89,6% (95/ 106)	86% (627/ 729)
WaldoInvestigation.StateMachine.GatheringModule	100% (2/ 2)	100% (4/ 4)	57,9% (33/ 57)

Figura 8.1: Waldo Investigation Unit Tests Results

Resultado Final: $\surd$

- **Análise**:

  As funcionalidades individuais de cada componente foram testadas, contudo à medida que o sistema cresce torna-se mais complexo de conseguir simular um estado para reproduzir uma falha que tenha ocorrido;

  Conseguiu-se uma boa confiança na estabilidade e interação entre os componentes do sistema;

  Há componentes que apresentam uma coverage mais baixa, contudo na sua maioria estes são scripts que são utilizados com pouca frequência e como tal a sua estabilidade é menos relevante;

  Os testes descritos nesta secção serão utilizados ao longo do tempo como testes de regressão.

8.1.2 ST02 - Teste à API para buscar dados do cliente do Waldo Web

- **Tipo de teste** - Teste de Coverage;

- **Objetivos** - Visto que a parte de buscar os dados do cliente é a mais importante do Waldo Web esta deve ser testada de modo a garantir que não entrega dados a quem não é suposto e ao mesmo tempo não apresenta nenhum comportamento fora do esperado.
- **Modelos** - get_client_data(request, client_name, data_type).

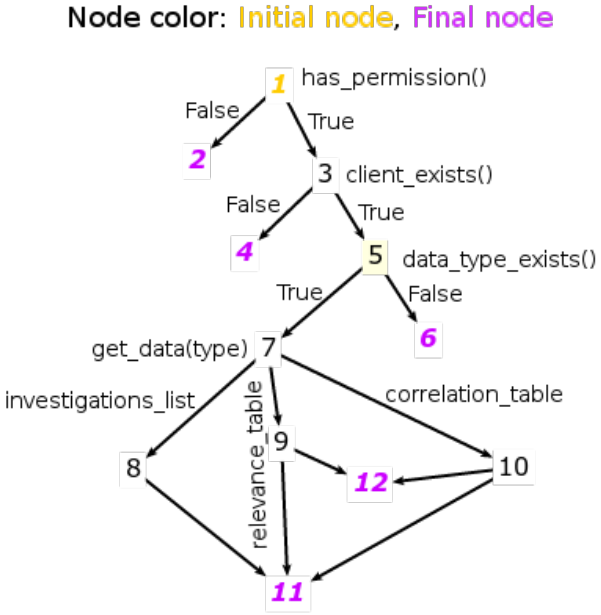


Figura 8.2: Client API Code Graph

- **Plano de Teste**:
  - Ferramentas:
    Graph Coverage.
  - Nivel de cobertura:
    Edge-Pair Coverage 100%.
  - Dominio de Teste:
    Test Requirement 1 - [1,2];
    Test Requirement 2 - [1,3,4];
    Test Requirement 3 - [1,3,5];
    Test Requirement 4 - [3,5,6];
    Test Requirement 5 - [3,5,7];
    Test Requirement 6 - [5,7,8];
    Test Requirement 7 - [5,7,9];
    Test Requirement 8 - [5,7,10];
    Test Requirement 9 - [7,8,11];

Test Requirement 10 - [7,9,11];
Test Requirement 11 - [7,9,12];
Test Requirement 12 - [7,10,11];
Test Requirement 13 - [7,10,12].

- **Casos de Teste**:
  - Test Case 1 - [1, 2];
  - Test Case 2 - [1, 3, 4];
  - Test Case 3 - [1, 3, 5, 6];
  - Test Case 4 - [1, 3, 5, 7, 9, 11];
  - Test Case 5 - [1, 3, 5, 7, 10, 11];
  - Test Case 6 - [1, 3, 5, 7, 8, 12];
  - Test Case 7 - [1, 3, 5, 7, 9, 12];
  - Test Case 8 - [1, 3, 5, 7, 10, 12].
- **Execução**:
  - Edge-Pair coverage obtida foi de 100%;
  - Ferramenta utilizada: unittest;
  - Casos de teste:

    O Test Case 1 foi obtido enviando um pedido sem ter um utilizador autenticado, deste resultou o código HTTP 302 que redireciona o utilizador para a página de login;

    O Test Case 2 foi obtido enviando um pedido de um utilizador autenticado, com um nome de cliente inválido. O sistema respondeu com o código HTTP 400 e notificando de ter recebido um cliente inválido;

    O Test Case 3 foi obtido enviando um pedido de um utilizador autenticado, com um nome de cliente válido, contúdo um tipo de dados pretendidos inválidos. O sistema respondeu com o código HTTP 400, e notificando que o tipo de dados pedidos não existem;

    O Test Case 4 foi obtido enviando um pedido de um utilizador autenticado, com um nome de cliente válido e com "relevance_table" como tipo de dados pretendidos, contudo este teste foi realizado sem ter nenhuns dados sobre a tabela de relevância na base de dados. O Sistema respondeu com o código HTTP 200, notificando que a tabela de relevância ainda não foi criada;

    O Test Case 5 foi obtido enviando um pedido de um utilizador autenticado, com um nome de cliente válido e com "correlation_table" como tipo de dados pretendidos, contudo este teste

foi realizado sem ter nenhuns dados sobre a tabela de correlação na base de dados. O Sistema respondeu com o código HTTP 200, notificando que a tabela de correlação ainda não foi criada;

O Test Case 6 foi obtido enviando um pedido de um utilizador autenticado, com um nome de cliente válido e com "investigations" como tipo de dados pretendidos, o servidor respondeu com o código HTTP 200 e com a lista de investigações em formato JSON;

O Test Case 7 foi obtido enviando um pedido de um utilizador autenticado, com um nome de cliente válido e com "relevance_table" como tipo de dados pretendidos, o servidor respondeu com o código HTTP 200 e com a a tabela de relevância em formato JSON;

O Test Case 8 foi obtido enviando um pedido de um utilizador autenticado, com um nome de cliente válido e com "correlatio_table" como tipo de dados pretendidos, o servidor respondeu com o código HTTP 200 e com a tabela de correlações em formato JSON.

- **Resultados de Teste**:
  - Foi obtido o nível de coverage pretendido;
  - Toda a API foi testada com sucesso, gerando respostas adequadas para cada um dos pedidos possíveis.
  - Resultado Final: $\surd$
- **Análise**:

  Agora possui-se confiança no funcionamento da API para buscar os dados do clientes, todos os comportamentos da mesma ficaram bem definidos e após quaisquer novas modificações podemos facilmente detetar se o funcionamento desta continua correto.

8.1.3 ST03 - Teste às funcionalidades do Sistema

- **Tipo de teste** - Teste de Casos de Uso;
- **Objetivos** - Simular ao nível do sistema, o comportamento dos casos associados a cada requisito funcional proposto inicialmente de modo a comprovar a correta implementação do sistema pretendido;
- **Modelos**:

Tabela 8.2: Requisitos Funcionais Implementados

ID	Requisito	Prioridade
RF03	**Receber logs**	MUST HAVE
RF01	**Analisar dados**	MUST HAVE
RF02	**Classificar alertas**	MUST HAVE
RF12	**Guardar investigações com alertas suspeitos**	MUST HAVE
RF04	**Apresentar Investigações**	MUST HAVE
RF05	**Aprender com feedback**	SHOULD HAVE
RF06	**Aumentar precisão de classificação**	SHOULD HAVE
RF11	**Apresentar fontes de apoio**	NICE TO HAVE

- **Plano de Teste**:
  - Ferramentas:
    JUnit4;
    unittest;
  - Todos os casos de teste devem ter um teste que passe;
  - Abordagem - Visto que são casos de uso, cada um destes vai ser descrito utilizando as categorias:
    * Ação;
    * Pré-requisitos;
    * Esperado;
    * Observado;
    * Casos Alternativos(Se existentes);
    * Resultado.
- **Casos de Teste**:
  - Test Case 1 - Receber dados dos IDS;
  - Test Case 2 - Interpretar dados recebidos dos IDS;
  - Test Case 3 - Avaliar um alerta recebido como sendo ou não uma intrusão;
  - Test Case 4 - Registar as investigações criadas na base de dados;
  - Test Case 5 - Visualizar as investigações registadas;
  - Test Case 6 - Classificar investigações na dashboard;
  - Test Case 7 - Comparar investigação em análise com as já existentes.
- **Execução**:

- Test Case 1:
  - Ação:
    Enviar dados recebidos no logstash provenientes dos vários IDS para a queue externa do Waldo Investigações.
  - Pré-requisitos:
    Setup dos IDS(Snort, Ossec e Bro);
    Setup do Logstash;
    Setup do pipeline dos vários IDS para o logstash;
    Setup do rabbitMQ;
    Setup da queue "externa" no rabbitMQ;
    Setup do pipeline do logstash para a queue "externa".
  - Esperado:
    O consumidor do Waldo Investigações associado à queue externa deteta uma nova mensagem na queue proveniente de um dos IDS e passa a mesma ao parser de alertas.
  - Observado: O consumidor do Waldo investigações associado à queue externa detetou uma nova mensagem na queue proveniente de um dos IDS recebeu e passou para o parser de alertas.
  - Resultado: $\surd$
- Test Case 2:
  - Ação:
    Parsar um alerta recebido, transformar num tipo de alerta e enviar para a queue do cliente a que está associado;
  - Pré-requisitos:
    Setup da queue "externa" no rabbitMQ;
    Setup da queue de cada cliente no rabbitMQ;
    Receber um alerta.
  - Esperado:
    Ao receber um alerta o consumidor vai entrega-lo ao parser, este por sua vez vai utilizar a fábrica de alertas e construir um objeto do tipo de alerta associado ao IDS que o enviou. Já com o alerta construido este vai ser enviado à queue do cliente associado ao mesmo.
  - Observado:
    Foi recebido um alerta do consumidor pelo parser, este por sua vez utilizou a fábrica de alertas e construiu um objeto do tipo de alerta associado ao IDS que o enviou. Já com o alerta construido este foi enviado à queue do cliente associado.
  - Casos Alternativos:

Foi recebido um alerta do consumidor pelo parser, este por sua vez utilizou a fábrica de alertas que detetou que o tipo de alerta recebido é inválido e cancelou o parse do alerta.

  * Resultado: $\surd$

- Test Case 3:
  * Ação:

    Receber alertas de forma constante até que apareça uma investigação na dashboard com a classificação dada à investigação.
  * Pré-requisitos:

    O servidor do Waldo Web tem de estar em funcionamento;

    O Waldo investigações precisa de estar em funcionamento e já ter o seu modelo estatístico definido;

    É necessário que estejam alertas a serem recebidos nas queues.
  * Esperado:

    Ao receber vários alertas o Waldo vai construido o seu modelo estatístico, se detetada uma variação em relação ao comportamento normal este vai despoletar a sua comparação com casos passados e caso ache suspeito regista como uma investigação. Esta investigação vai aparecer na dashboard de investigações do Waldo Web com uma classificação atribuída.
  * Observado:

    Foram recebidos vários alertas do sistema em produção e passado alguns minutos apareceu uma investigação na dashboard de investigações.
  * Resultado: $\surd$
- Test Case 4:
  * Ação:

    Instrumentar o código para notificar quando detetar algum alerta suspeito. Quando for recebida a notificação vamos verificar se esta foi registada na base de dados de investigações.
  * Pré-requisitos:

    A base de dados de postgreSQL já precisa estar construida e configurada;

    O Waldo investigações precisa de estar em funcionamento.
  * Esperado:

    Será lançado um aviso de que algo suspeito foi detetado e ao verificar as investigações na base de dados teremos lá uma nova investigação com os dados suspeitos encontrados.

* Observado:

  Quando apresentado o aviso de que um comportamento suspeito foi detetado, foi analisada a base de dados e viu-se que uma nova investigação tinha sido criada.

* Resultado: $\surd$

- Test Case 5:
  * Ação:

    Aceder à interface do Waldo Web e observar uma investigação.

  * Pré-requisitos:

    O servidor do Waldo Web tem de estar em funcionamento;

    Já precisam de estar investigações registadas na base de dados.

  * Esperado:

    A lista de investigações é apresentada e ao carregar no botão para visualizar a investigação esta é apresentada.

  * Observado:

    A lista de investigações foi apresentada e ao carregar no botão para visualizar a investigação foi apresentada com todos os seus elementos.

  * Resultado: $\surd$

- Test Case 6:
  * Ação:

    Instrumentar o código para notificar se for recebido feedback e quando for aplicado a algo. Avaliar uma investigação das apresentadas na interface do Waldo Web.

  * Pré-requisitos:

    O servidor do Waldo Web tem de estar em funcionamento;

    Já precisam de estar investigações registadas na base de dados;

    O Waldo investigações precisa de estar em funcionamento.

  * Esperado:

    Será observada uma investigação por avaliar e será feita a sua avaliação despoletando o envio de feedback para o Waldo Investigações que por sua vez irá aplicar este feedback no seu sistema de conhecimento baseado em casos e notificar quando o fizer.

  * Observado:

    Foi realizada a avaliação de uma investigação na dashboard web despoletando o envio de feedback para o Waldo

Investigações, este por sua vez aplicou este feedback no sistema de conhecimento baseado em casos.

    * Resultado: $\surd$

  - Test Case 7:

    * Ação:

      Avaliar investigações e observar as investigações geradas.

    * Pré-requisitos:

      O servidor do Waldo Web tem de estar em funcionamento;

      Já precisam de estar investigações registadas na base de dados;

      O Waldo investigações precisa de estar em funcionamento.

    * Esperado:

      Após avaliar várias investigações e ainda estar dentro da mesma janela temporal consultamos as investigações mais próximas das novas investigações a serem geradas, notando a influência do feedback das ultimas investigações avaliadas nas novas investigações a serem geradas.

    * Observado:

      Após a avaliação de várias investigações na janela temporal atual notou-se que as novas investigações mais próximas das novas investigações a serem geradas sofreram influência do feedback fornecido.

    * Resultado: $\surd$

- **Resultados de Teste**:

  - Foram testados todos os requisitos implementados;
  - Todos os testes aos requisitos implementados coincidiram com o resultado esperado.
  - Resultado Final:

Tabela 8.3: Requisitos Funcionais Implementados

ID	Requisito	Prioridade	Resultado
RF03	**Receber logs**	MUST HAVE	√
RF01	**Analisar dados**	MUST HAVE	√
RF02	**Classificar alertas**	MUST HAVE	√
RF12	**Guardar investigações com alertas suspeitos**	MUST HAVE	√
RF04	**Apresentar Investigações**	MUST HAVE	√
RF05	**Aprender com feedback**	SHOULD HAVE	√
RF06	**Aumentar precisão de classificação**	SHOULD HAVE	√
RF11	**Apresentar fontes de apoio**	NICE TO HAVE	√

- **Análise**:
  - Com este teste exercitamos cada uma das funcionalidades propostas e comprovamos que os objetivos de cada uma delas foram atingidos.

8.2 Testes dos Requisitos Não Funcionais

8.2.1 ST04 - Aumentar precisão de classificação ao longo do tempo

- **Tipo de teste**:

  Use case testing.

- **Objetivos**:

  Lançar menos investigações que sejam desconhecidas e ir melhorando as classificações das investigações já conhecidas.

- **Modelos**:

  O sistema deve comportar-se segundo o modelo apresentado na figura 8.3.

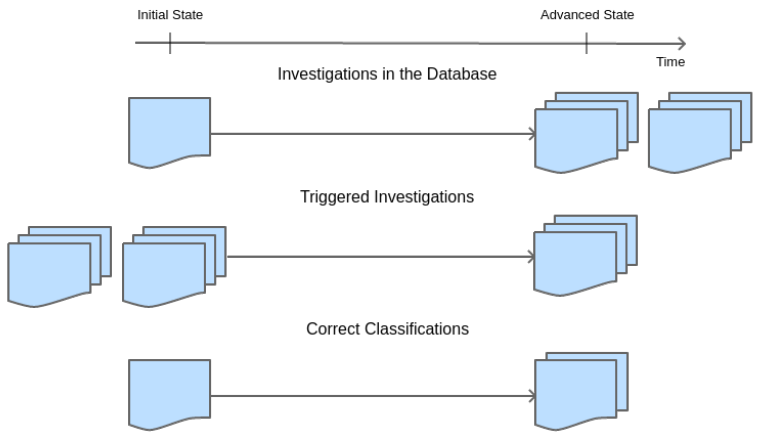


Figura 8.3: Progress of investigations over time

- **Plano de Teste**:

  Reduzir quantidade de investigações despoletadas por distância aos casos de ataque já conhecidos;

  Classificar novas investigações com base em casos de ataque já classificados pelos operadores.

- **Casos de Teste**:

  Caso de teste 1 - Partindo de um estado limpo do sistema, comparar a quantidade de investigações geradas no primeiro dia com a quantidade de investigações após o sistema estar em funcionamento durante 5 dias seguidos;

  Caso de teste 2 - Partindo de um estado limpo do sistema, comparar as classificações dadas pelo Waldo no primeiro dia com as classificações de investigações após o sistema estar em funcionamento durante 5 dias seguidos.

- **Execução**:

  Foram observadas as investigações despoletadas pelo waldo, observando os alertas de cada categoria apresentada e realizando a sua classificação durante 1 semana;

  Foram realizadas queries à base de dados de investigações para obter a quantidade de investigações a cada dia, o total de investigações e a quantidade de investigações corretamente classificadas.

- **Resultados de Teste**:

– Caso de teste 1:

Quantidade de investigações geradas no primeiro dia	63
Quantidade de investigações geradas no quinto dia	49

– Caso de teste 2:

investigações geradas no primeiro dia	Todas lançadas como "Not Attack"
investigações geradas no quinto dia	A maioria é lançada como "Not Attack", mas por vezes já são classificadas como "Attack" investigações próximas a ataques anteriormente classificados

– Resultado Final: $\surd$

- **Análise**:

  A quantidade de investigações diminuiu ao longo do tempo e a classificação das investigações foi melhorando;

  Atingiu-se o objetivo pretendido, contudo mesmo notando melhoramentos ao longo do tempo, este processo de aprendizagem deveria ser muito mais longo de modo a conseguir observar resultados mais acentuados.

8.2.2 ST05 - Precisão alta de classificação de alertas

- **Tipo de teste**:

  Use case testing.

- **Objetivos**:

  Perceber se o sistema consegue classificar alertas com boa precisão.

- **Modelos**:

  Dados a recolher no fim do teste:

Descrição	**Total**
Investigações lançadas	?
Investigações classificadas como "Not Attack"	?
Investigações classificadas como "Attack"	?
Investigações corretamente classificadas como "Not Attack"	?
Investigações corretamente classificadas como "Attack"	?
Accuracy	?
Incidentes detetados	?
Incidentes detetados pelos operadores	?
Incidentes detetados pelo Waldo	?
Precisão	?

- **Plano de Teste**:

  Precisão acima de 50%;

  Simular o estado da rede em que se sabe que houve um incidente e verificar se o Waldo detetou.

- **Casos de Teste**:

  Caso de teste 1 - Começando com o sistema vazio, receber dados dos vários IDS em produção durante uma semana e classificar investigações juntos com os operadores.

- **Execução**:

  Foram observadas as investigações despoletadas pelo waldo, observando os alertas de cada categoria apresentada e realizando a sua classificação durante 1 semana;

  Foram realizadas queries à base de dados de investigações para obter a quantidade de investigações a cada dia, o total de investigações e a quantidade de investigações corretamente classificadas.

- **Resultados de Teste**:

  Durante a semana foram detetados 2 incidentes pelo Waldo e pela equipa foram detetados outros 2 incidentes, totalizando 4 incidentes;

  Dos dois incidentes detetados pelo Waldo ambos foram classificados como "Not Attack" pelo sistema CBR;

Descrição	**Total**	**Percentagem**
Investigações lançadas	240	100%
Investigações classificadas como "Not Attack"	231	96,25%
Investigações classificadas como "Attack"	9	3,75%
Investigações corretamente classificadas como "Not Attack"	229	99,13%
Investigações corretamente classificadas como "Attack"	0	0%
Accuracy	229/240	95,42%
Incidentes detetados	4	100%
Incidentes detetados pelos operadores	2	50%
Incidentes detetados pelo Waldo	2	50%
Precisão	2/4	50%

  Resultado Final: $\surd$

- **Análise**:

  Obtivemos uma accuracy de 95,42%;

  Neste caso especifico a accuracy obtida deve-se ao facto de o waldo determinar a situação como suspeita, criando uma investigação com a classificação "Not Attack", sempre que encontra algo fora dos modelos que tem registados na base de dados de investigações. Como na

maioria das vezes não há nenhum problema, as classificações estão corretas;

Mesmo assim este teste permitiu-nos a validação do Waldo na sua capacidade de detetar comportamentos suspeitos. Observando as investigações foram encontrados vários comportamentos suspeitos e duas das investigações detetaram incidentes;

O Waldo mostrou capacidade de análise continua ao comportamento da rede, este encontrou problemas que muito dificilmente os operadores iriam notar com o seu modo de análise regular, notando-se isto nos incidentes detetados pelos operadores pois estes eram diferentes dos detetados pelo Waldo;

A deteção de comportamentos suspeitos funcionou bem, contudo de modo a obter classificações mais corretas e diminuir o tempo necessário de análise de investigações, necessita-se de um maior tempo de treino e avaliação para que este passe a reconhecer mais casos possíveis e estados da rede.

8.2.3 ST06 - Automatizar tarefas

- **Tipo de teste**:

  Use case testing.

- **Objetivos**:

  Reduzir tempo de tarefas repetidas.

- **Modelos**:

Tempo de primeira análise de categoria de alerta	?
Tempo de segunda análise a categoria de alerta	?

- **Plano de Teste**:

  - Ferramentas:

    Gnome Clocks.

  - Após uma análise a um tipo de categoria, reduzir o tempo que será necessário para a próxima análise à mesma categoria.

- **Casos de Teste**:

  Caso de teste 1 - Vai ser realizada uma primeira análise a uma categoria em que vão ser registados os sítios que forem pesquisados e de seguida vai ser passada outra categoria já com links de uma análise passada.

- **Execução**:

  Caso de teste 1 - Foram realizados os passos:

1. Foi apresentada uma investigação a um operador;
2. Foram escolhidos dois nós das investigação que possuíam diferentes categorias com 3 alertas cada uma.
3. A categoria do primeiro nó (bro_noticelog) não possuía links associados de pesquisas anteriores, enquanto a categoria do segundo nó (A Network Trojan Was Detected) possui já links de pesquisas anteriores;
4. Quando o operador abriu a página de cada categoria foi colocado o cronómetro a contar e quando percebeu o significado da categoria e dos alertas que foram gerados foi parado.

- **Resultados de Teste**:

Tempo de primeira análise de categoria de alerta	5m 57s
Tempo de segunda análise a categoria de alerta	3m 48s

Resultado Final: $\surd$

- **Análise**:

Com a implementação dos links associados às categorias obter-se uma maior facilidade na capacidade de encontrar informação sobre dados associados às categorias já investigadas.

8.2.4 ST07 - Reduzir tempo gasto desde a identificação do alerta até à resolução do incidente

- **Tipo de teste**:

Use case testing.

- **Objetivos**:

Perceber se o sistema construído realmente apoia os operadores na resolução de incidentes.

- **Modelos**:

Estimativa de tempo médio de resolução de incidente	30min
Tempo médio de resolução de incidente utilizando Waldo	?

- **Plano de Teste**:
  - Ferramentas:

    Gnome Clocks.
  - Utilizando o Waldo pretende-se obter um tempo médio até à resolução do incidente mais baixo que o tempo médio normal.

- **Casos de Teste**:

  Caso de teste 1 - Avaliar três investigações que espoletem incidentes e verificar se o seu tempo médio de resolução é mais baixo que o tempo normal;

- **Execução**:

  - Caso de teste 1:

    Foram analisadas investigações continuamente, sempre que se iniciava a análise de uma investigação ativou-se o cronómetro;

    Quando estas investigações realmente detetam um incidente, este é analisado e quando este é resolvido guarda-se o tempo demorado;

    Foi registado o tempo de resolução de três incidentes;

- **Resultados de Teste**:

Estimativa de tempo médio de resolução de incidente	30min
Tempo médio de resolução de incidente utilizando Waldo	14m 48s

  Resultado Final: $\surd$

- **Análise**:

  Visto que a investigação apresenta directamente o problema, rapidamente se conseguir ir desde a análise da investigação até à causa do problema;

8.2.5 ST08 - Adaptabilidade a várias fontes diferentes

- **Tipo de teste**:

  Use case testing.

- **Objetivos**:

  Mostrar que são necessárias poucas ou nenhumas modificações ao código já existente para adicionar uma nova fonte de informação.

- **Modelos** - Nova fonte de dados 8.4:

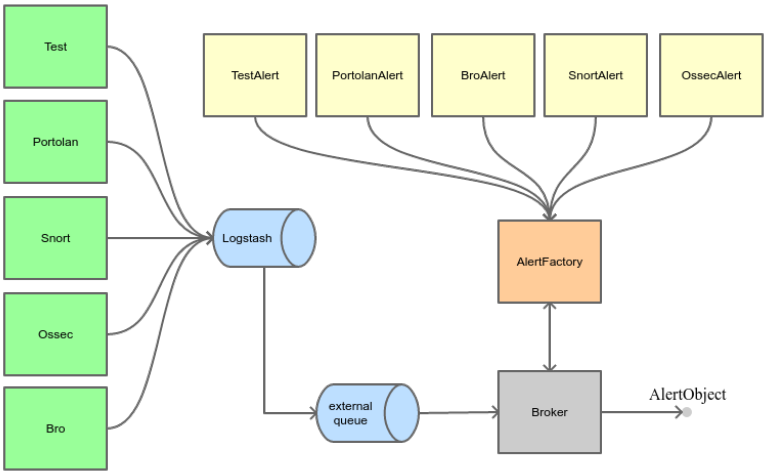


Figura 8.4: Adicionar nova fonte de dados

- **Plano de Teste**:

  Pretende-se a criação de um novo tipo de alerta e o seu tratamento com um mínimo numero de alterações.

- **Casos de Teste**:

  Caso de Teste 1 - Testar complexidade de criação e integração de um novo alerta no parse inicial dos alertas;

  Caso de Teste 2 - Testar comportamento do sistema ao processar alertas do tipo TestAlert.

- **Execução**:

  - Ferramentas:

    Intellij IDEA;

    Git.

  - Caso de Teste 1 - Para construir um novo tipo de alerta foram realizados os passos:

    1. Adicionar tipo TEST na enumeração AlertType;
    2. Criar classe TestAlert que estende a classe pai Alert;
    3. Definir o tipo do alerta como TEST no construtor;
    4. Definir a nome do parâmetro que indica qual a categoria do alerta na categoria;

5. Adicionar nova possibilidade na AlertFactory para construir a classe TestAlert se o tipo recebido na mensagem que vem da queue possuir o tipo "test".

– Caso de Teste 2 - Para testar o comportamento do sistema com um novo alerta foram realizados os passos:

1. Criar um ficheiro de texto para colocar os alertas a enviar;
2. Associar este ficheiro ao logstash como uma fonte de dados utilizando o plugin de json para parsar os dados;
3. Selecionar alertas de outros IDS e modificar o seu tipo para "test";
4. Enquanto o sistema corre adicionar esses alertas ao ficheiro;
5. Verificou-se o funcionamento normal do sistema.

- **Resultados de Teste**:

  – Foram necessárias poucas modificações ao código que já existia, como se pode ver na figura 8.5.

```
diff --git a/WaldoInvestigation/src/WaldoInvestigation/Alerts/AlertFactory.java b/WaldoInvestigation/src/WaldoInvestig
ation/Alerts/AlertFactory.java
index 20bbc94..35754f1 100644
--- a/WaldoInvestigation/src/WaldoInvestigation/Alerts/AlertFactory.java
+++ b/WaldoInvestigation/src/WaldoInvestigation/Alerts/AlertFactory.java
@@ -35,6 +35,10 @@ public class AlertFactory {
         {
             alert = new PortolanAlert(json);
         }
+        else if(type.contains("test"))
+        {
+            alert = new TestAlert(json);
+        }
         else
         {
             try {
diff --git a/WaldoInvestigation/src/WaldoInvestigation/Alerts/AlertType.java b/WaldoInvestigation/src/WaldoInvestigati
on/Alerts/AlertType.java
index f310544..490977c 100644
--- a/WaldoInvestigation/src/WaldoInvestigation/Alerts/AlertType.java
+++ b/WaldoInvestigation/src/WaldoInvestigation/Alerts/AlertType.java
@@ -6,5 +6,5 @@ package WaldoInvestigation.Alerts;
  * PURPOSE: Definition of the available Alert Types
  **/
 public enum AlertType {
-    SNORT, OSSEC, BRO, PORTOLAN
+    SNORT, OSSEC, BRO, PORTOLAN, TEST
 }
```

Figura 8.5: Modificações no código já existente para novo alerta

  – Resultado Final: $\surd$

- **Análise**:

  – Com este teste validou-se que é fácil registar novos alertas não sendo necessárias quase modificações ao código já existente;

  – Os novos alertas ficaram directamente assocaidos aos modelos estatisticos e novas investigações;

  – A implementação dos alertas é também esta simples visto que a sua funcionalidade base é implementada estendendo a classe Alert e definindo apenas o tipo de Alerta da classe de alerta e como se busca a categoria do JSON recebido;

– Deste modo ganhou-se confiança na simplicidade de adicionar novas fontes e do rápido impacto destas no sistema;

8.2.6 ST09 - Modularidade

- **Tipo de teste**:

  Use case testing.

- **Objetivos**:

  Mostrar que são necessárias poucas ou nenhumas modificações ao código já existentes para adicionar um novo módulo de correlacionamento/classificação.

- **Modelos**:

  Posição de adição de novos módulos 8.6;

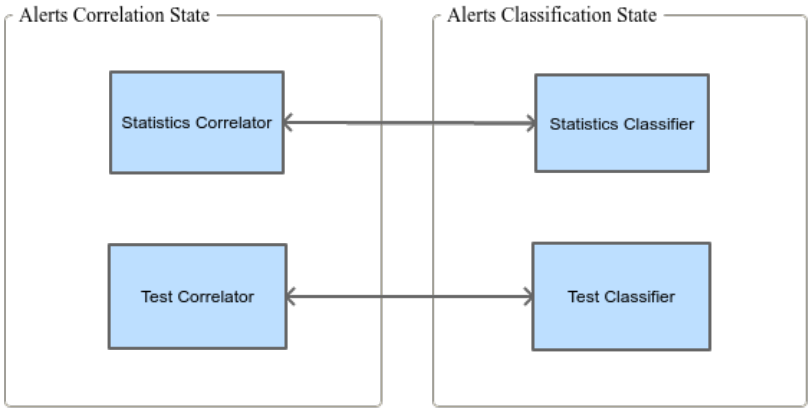


Figura 8.6: Independência entre os pares Correlacionador/Classificador

- **Plano de Teste**:

  – Não deve ser necessário modificar nenhuma lógica das classes já existentes para a implementação de um novo módulo, apenas podemos realizar modificações em zonas que vão inicializar o módulo, e enumerações para especificar os tipos de módulos existentes.

- **Casos de Teste**:

- Caso de teste 1 - Testar a dificuldade de construir e integrar um par correlacionador/classificador no Waldo Investigações;
- Caso de teste 2 - Testar a troca de informação do correlacionador para o classificador durante a execução do Waldo.

- **Execução**:
  - Ferramentas:

    Intellij IDEA;

    Git.
  - Caso de teste 1 - Passos realizados:
    1. Criação de classe TestCorrelator que estende a classe abstrata Correlator;
    2. Implementação dos métodos abstratos da classe pai e do método construtor;
    3. Modificação de isCorrelatorRunning para false no fim da execução do mesmo;
    4. Criação de classe TestClassifier que estende a classe abstrata Classifier;
    5. Implementação dos métodos abstratos da classe pai e do método construtor;
    6. Modificação de isClassifierRunning para false no fim da execução do mesmo;
    7. Adição do tipo TEST na enumeração InvestigationType;
    8. Registo do TestCorrelator nos correlacionadores da máquina de estados durante a sua inicialização;
    9. Registo do TestClassifier nos classificadores da máquina de estados durante a sua inicialização.
  - Caso de teste 2 - Passos realizados:
    1. Implementação resultante do teste 1;
    2. Criação e registo de um par chave/valor de duas strings com o valor "Hello" e "World" no TestCorrelator;
    3. Registo deste par chave/valor nos resultados do correlacionador;
    4. Implementação da busca do resultado dos correlacionador no TestClassifier;
    5. Imprimir resultado para a consola.

- **Resultados de Teste**:

  Ao código já existente foram necessárias poucas modificações, sendo estas realizadas na inicialização da máquina de estados e na enumeração dos tipos de investigação como se pode ver na figura 8.7;

Figura 8.7: Modificações no código já existente

A passagem dos resultados do classificador para o correlacionar funcionaram sem problemas;

Resultado Final: $\surd$

- **Análise**:

  Com este teste validou-se que é fácil registar e remover pares de correlacionadores/classificadores, ganhando-se confiança na solução arquitetural quanto à sua modularidade.

  A implementação de um correlacionador e classificador foi simples visto que já existiam as classes pai que possuem o comportamento base completo necessário;

  Foi necessário manualmente a modificação dos valores isCorrelatorRunning e isClassifierRunning no fim da execução das tarefas do correlacionar e classificador. Visto que esta modificação ocorre sempre no fim de cada classificador e correlacionador esta modificação foi implementada nas classes pai de modo a ser automaticamente ativada no fim da execução das tarefas dos filhos.

8.2.7 ST10 - Perfomance

Para validar que a performance atinge os requisitos definidos na secção 2.5 foi realizado o seguinte teste.

- **Tipo de Teste**: Black-box - State Transition Testing;
- **Objetivos**: Entender se o sistema consegue realizar as investigações de alertas dentro do tempo definido no seu uso regular;
- **Modelos**: Diagrama da máquina de estados incluindo a instrumentação para contar o tempo demorado 8.8;

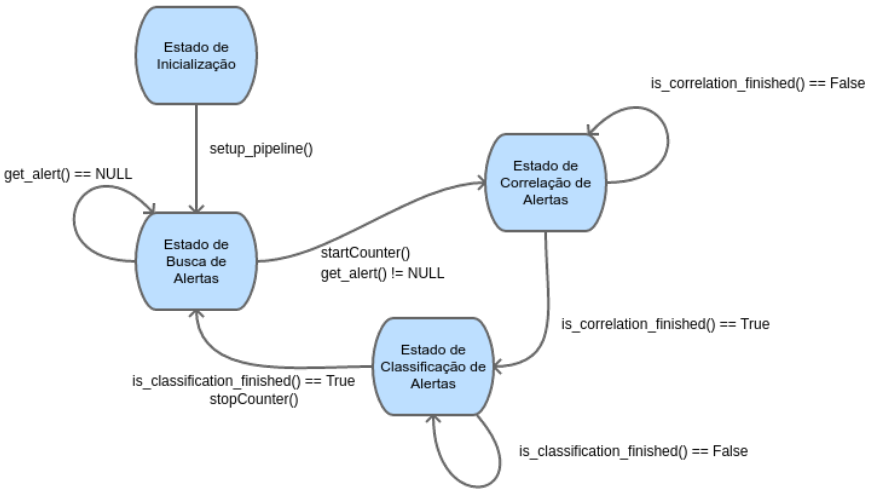


Figura 8.8: Estados de Investigação com chamadas de contagem do tempo

- **Plano de Teste**:

  Estratégia

  - Será simulado o uso normal do módulo de investigações enquanto medimos o tempo médio que este leva desde que é lido um alerta no AlertGatheringState, até ao momento em que é registado o resultado do AlertClassificationState;
  - Será medido o uso do cpu e a memória consumida de modo a possuirmos visibilidade sobre a possibilidade de bottlenecks por parte da máquina a usar para o teste;
  - O tempo médio deve ser abaixo de 1s.

  Ferramentas

  - htop[96] - Para visualização da utilização de cada core do CPU e visualização da memória utilizada por cada processo sem buffers;
  - top[97] - Para visualização da utilização de memória utilizada por cada processo incluindo buffers.

- **Casos de Teste**: Iremos executar o módulo de investigações do Waldo durante 1 dia completo com uma janela temporal de 1h no ambiente real, recebendo os alertas diretamente do logstash com os vários IDS da organização.

- **Execução**:

  - Para garantir que o teste começa num estado fixo, foi limpa a base de dados e começou-se sem dados nenhuns disponíveis;
  - Foi adicionado um contador que começa a contar quando é recolhido o alerta no estado AlertGatheringState e este é parado quando termina o estado AlertCorrelationState, devolvendo o tempo que passou em milésimos de segundo(ms);
  - Este valor é acumulado e é apresentada a sua média a cada 20 alertas investigados (20 ciclos dos estados);
  - A visualização do cpu e da ram utilizada é realizada continuamente com as ferramentas htop e top;

- **Resultados de Teste**:
  - O sistema apresentou um tempo médio de investigação de alertas de 965ms, estando abaixo do intervalo de tempo máximo definido;
  - Notou-se uma utilização do cpu por picos, estando na maioria do tempo cada um dos cores entre os 20% e 40% de uso;
  - A memória em utilização ao fim de um dia foi de 1379MB;
  - Resultado Final: $\surd$

- **Análise**:
  - O tempo de investigação de cada alerta foi dentro do espaço temporal definido, contudo com o aumento de hiper alertas e casos de ataque acumulados este passado mais uns dias poderia ultrapassar o limite de 1s;
  - Obteve-se sucesso no que é a parte mais importante que era conseguir analisar todos os alertas em tempo útil;
  - Notou-se contudo um perigoso aumento constante de memória utilizada, este num maior espaço de tempo iria acabar por esgotar a memória da máquina.

Considerando os resultados obtidos, percebeu-se que o aumento constante de memória vem da acumulação de hyper alertas no correlacionador baseado em estatística para permitir o recalculo das tabelas de relevância e correlação. Este problema além de esgotar a memória da máquina vai também influenciar a performance pois vai forçar uma maior quantidade de cálculos para obter novas tabelas à medida que a quantidade de hiper alertas aumente. Para mitigar este problema foram realizadas as implementações apresentadas na secção 7.3.1 e para comprovar as vantagens desta nova implementação foi repetido o teste de performance.

- **Tipo de teste** - O mesmo que acima;

- **Objetivos** - O mesmo que acima;
- **Modelos** - O mesmo que acima;
- **Plano de Teste** - O mesmo que acima;
- **Casos de Teste** - O mesmo que acima;
- **Execução** - O mesmo que acima;
- **Resultados de Teste**:
  - O sistema apresentou um tempo médio de investigação de alertas de 301ms, estando abaixo do intervalo de tempo máximo definido;

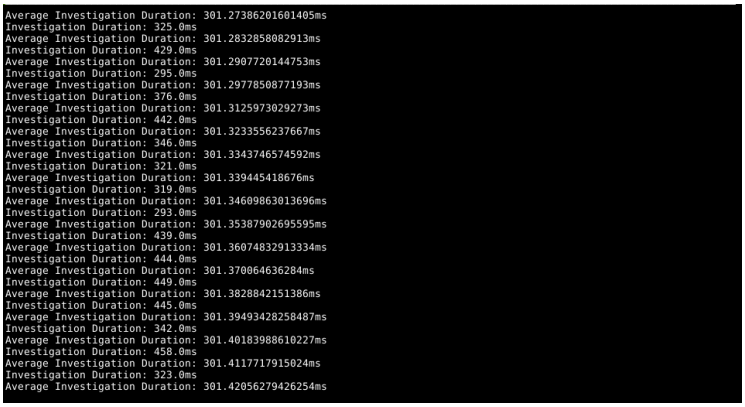


Figura 8.9: Tempo de análise médio de alertas após modificações

  - Notou-se uma utilização do cpu apenas quando recebidos alertas, estando esta em média entre os 35% a 50%.
  - A memória em utilização ao fim de um dia foi de 620MB;

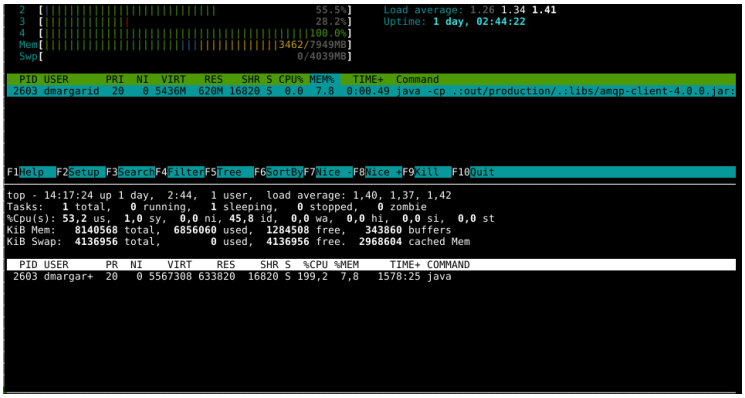


Figura 8.10: Utilização de cpu e ram após modificações

  - Resultado Final: $\surd$

- **Análise** - Foram obtidos grandes melhoramentos reduzindo a quantidade de dados armazenados, ganhou-se em tempo de processamento visto que são necessárias menos inserções e houve uma grande diminuição de dados em memória, reduzindo para menos de metade os dados armazenados durante um dia e mantendo o crescimento ao longo do tempo mais reduzido que antes.

8.3 Conclusão

Com este capitulo encontraram-se falhas no sistema que foram corrigidas e melhoraram o sistema como um todo. Ao mesmo tempo que se deu o melhoramento do sistema também se ganhou confiança nas suas funcionalidades.

Capítulo 9

Conclusão

Dado que foram feitas conclusões especificas em cada capítulo, aqui é feita uma conclusão mais geral e é dado como terminado o projeto apresentando um resumo do que foi feito, algumas reflexões e possível trabalho para futuro. Considerando os objetivos do projeto, conseguimos com a construção do sistema:

- Modelar o estado normal da rede de um cliente através da correlação de eventos de segurança de várias fontes;
- Detetar comportamentos suspeitos na rede dos clientes de modo autónomo;
- Comparar o estado atual da rede e comportamentos suspeitos detetados com eventos passados similares;
- Classificar as investigações construidas a partir de comportamentos suspeitos;
- Apresentar investigações, numa dashboard simples e elegante, a operadores demonstrando as correlações de informação feitas pelo Waldo de forma estruturada para simplificar a sua análise;
- Registar sítios visitados quando resolvida uma investigação e apresentação desses mesmos sítios em novas investigações parecidas;
- Moldar ao longo do tempo a aprendizagem da plataforma com a utilização do feedback do operador;

9.1 Reflexões

Este documento representa o esforço, empenho e dedicação ao longo de dois semestres de estágio.

Aqui pode ver-se de forma clara como foi construído o sistema. Começou-se por verificar os objetivos e perceber os requisitos pretendidos à implementação do sistema, foram estudados produtos do mercado que se pudessem assemelhar ao Waldo, foram assimilados artigos sobre técnicas que poderiam ajudar na implementação, de seguida foi realizado um planeamento temporal para os requisitos previamente definidos, com base nos requisitos e no seu planeamento foi definida uma arquitetura que garanta um produto de qualidade, escolhidas ferramentas para esta, definida uma metodologia para o modo de trabalhar e gerir o projeto, seguiu-se a implementação do sistema e a sua validação consoante os requisitos funcionais e não funcionais definidos.

Houve um desvio do planeamento definido, contudo todos os requisitos *Must Have* foram implementados, e a maioria dos *Should Have* também. Este desvio deu-se devido à complexidade dos correlacionadores e classificadores que levaram ao atraso no seu desenvolvimento, contudo como já previsto nos riscos apresentados no capitulo de planeamento, foram implementadas primeiro as partes mais perigosas e complexas do sistema. Atendendo aos resultados dos testes efetuados, pode-se concluir que foram cumpridos os objetivos do estágio.

Foi obtido um sistema capaz de a partir de várias fontes indefinidas de alertas definir um modelo estatístico do seu comportamento normal e que percebe quando existem variações deste. Ao mesmo tempo que este percebe estas variações ainda as compara com as bases de conhecimento de casos passados fornecidas pelos operadores de segurança realizando uma avaliação deste casos suspeitos como um perito. Este sistema ainda permite a fácil inserção de novos correlacionadores e classificadores fornecendo já a infraestrutura para receção e tratamento de alertas, separação destes por vários clientes, simulando a individualidade da rede de cada cliente, oferece a funcionalidade base para a criação de investigações e também uma interface para a apresentação destas.

Este projeto foi um desafio constante, sendo um produto diferente de todos os apresentados no capitulo 3 e visto que o sistema teve como base vários artigos que apresentam apenas abordagens experimentais. Deste modo existiu uma grande necessidade de conseguir modelar o sistema baseado apenas nos conceitos apresentados, adaptar o sistema em relação ao ambiente do autor e integrar e modificar os vários conceitos de modo a construir um sistema único que tire vantagem de cada uma das abordagens enquanto cumpre com todos os requisitos definidos. Outra grande fonte de aprendizagem foi a comunicação e gestão de requisitos junto do cliente(a empresa) e das várias partes interessadas, o planeamento de um projeto que é bastante arriscado e a sua gestão de riscos e objetivos que precisa de ter associada uma grande capacidade de resposta rápida e eficaz.

Neste momento, o Waldo encontra-se em produção, disponível para ser utilizado pelos operadores, analisando continuamente os dados da rede da

Dognædis e gerando investigações consoante os comportamentos suspeitos na rede.

9.2 Trabalho Futuro

Os próximos passos a tomar que podem fazer o sistema evoluir serão:

- Fornecer mais avaliações para a base de casos de ataque de modo a permitir que o sistema consiga avaliar situações de forma mais correta;
- Observar os comportamentos detetados pelo Waldo e ajustar os thresholds da plataforma de modo a maximizar a qualidade da informação apresentada ao mesmo tempo que se tenta reduzir a quantidade de ruído;
- Instalar o Waldo nas redes dos clientes de modo a conseguir visualizar o estado das redes destes e encontrar possíveis ameaças;
- Implementar um novo conjunto de correlacionador/classificador com base em semelhanças como apresentado no artigo descrito em 3.2.4;
- Integrar os resultados de novos pares de modo a obter mais certeza sobre cada investigação;
- Com a implementação de novos pares faz sentido também existirem novas formas de visualização dos diferentes resultados, provendo mais conhecimento sobre o próprio funcionamento da rede;
- A integração com a plataforma de inteligência da empresa de modo a melhorar os resultados apresentados em cada investigação apresentando estes com fontes suspeitas associadas.

Bibliografia

[1] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," *Proceedings of the IEEE Symposium on Security and Privacy*, 2010.

[2] V. Y. M. F. Guofei Gu, Phillip Porras and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation," *16th USENIX Security Symposium (Security'07)*, 2007.

[3] J. R. Mirheidari SA., Arshad S., "Alert correlation algorithms: A survey and taxonomy," *Cyberspace Safety and Security*, pp. 83–97, 2013.

[4] M. D. Reuben Smith, Nathalie Japkowicz and P. Mason, "Using unsupervised learning for network alert correlation," *Advances in Artificial Intelligence*, pp. 308–319, 2008.

[5] N. S. Hanli Ren and A. A. Ghorbani, "An online adaptive approach to alert correlation," *DIMVA'10*, p. 153–172, 2010.

[6] W. Z. A. Zakaria, "Application of case based reasoning in it security incident response," *ICRET*, 2015. Acedido a 27 de Janeiro de 2017.

[7] "Broker." `https://msdn.microsoft.com/en-us/library/ff648096.aspx`. Acedido a 06 de Agosto de 2017.

[8] "Cross-validation definition." `https://www.cs.cmu.edu/~schneide/tut5/node42.html`. Acedido a 2 de Dezembro de 2016.

[9] "Ears - the easy approach to the requirements syntax." `http://www.iaria.org/conferences2013/filesICCGI13/ICCGI_2013_Tutorial_Terzakis.pdf`. Acedido a 23 de Novembro de 2016.

[10] "Istqb - foundation level syllabus." `http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4`. Acedido a 23 de Novembro de 2016.

[11] "Gnu/linux." `https://www.gnu.org/gnu/linux-and-gnu.pt-br.html`. Acedido a 26 de Dezembro de 2016.

[12] "Overfitting definition." https://en.wikipedia.org/wiki/Overfitting. Acedido a 2 de Dezembro de 2016.

[13] "Precision and recall definition." https://en.wikipedia.org/wiki/Precision_and_recall. Acedido a 29 de Novembro de 2016.

[14] "Root definition." http://www.linfo.org/root.html. Acedido a 21 de Novembro de 2016.

[15] "What is security orchestration." https://blog.komand.com/what-is-security-orchestration. Acedido a 8 de Dezembro de 2016.

[16] "Dognaedis." https://www.dognaedis.com/. Acedido a 9 de Abril de 2015.

[17] "Isec." https://www.isec.pt/PT/Instituto/. Acedido a 20 de Julho de 2017.

[18] "Codev." https://codev.dognaedis.com/. Acedido a 9 de Abril de 2015.

[19] "i* concepts." https://www.cs.toronto.edu/~gross/istar/indexd.html. Acedido a 23 de Novembro de 2016.

[20] E. S. K. Yu and J. Mylopoulos, "Understanding "why" in software process modelling, analysis, and design," *Proc. 16th Int. Conf. on Software Engineering*, pp. 159–168, 1994.

[21] E. S. K. Yu and J. Mylopoulos, "Modelling organizational issues for enterprise integration," *ICEIMT'97 - International Conference on Enterprise Integration and Modeling Technology. Turin, Italy*, pp. 529–538, 1997.

[22] "The open source elastic stack." https://www.elastic.co/products. Acedido a 21 de Novembro de 2016.

[23] "Hexadite." https://www.hexadite.com/. Acedido a 8 de Dezembro de 2016.

[24] "Splunk." https://www.splunk.com. Acedido a 9 de Dezembro de 2016.

[25] "Fireeye." https://www.fireeye.com/analyze-and-respond.html. Acedido a 12 de Dezembro de 2016.

[26] "Neural network follies." https://neil.fraser.name/writing/tank/. Acedido a 13 de Dezembro de 2016.

[27] "Snort." https://www.snort.org/. Acedido a 18 de Novembro de 2016.

[28] N. J. a. D. Reuben Smith, "Clustering using an autoassociator: A case study in network event correlation," *PDCS 2005*, pp. 613–618, 2005.

[29] "Malaysia computer emergency response team." https://www.mycert.org.my/en/. Acedido a 30 de Janeiro de 2017.

[30] R. R. Bharat Rao, Glenn Fung, "On the dangers of cross-validation. an exprimental evaluation," *SDM*, pp. 588–596, 2008.

[31] "Ossec - open source hids security." http://ossec.github.io/. Acedido a 17 de Novembro de 2016.

[32] "Snort: The world's most widely deployed ips technology." http://www.cisco.com/c/en/us/products/collateral/security/brief_c17-733286.html. Acedido a 21 de Novembro de 2016.

[33] "The bro network security monitor." https://www.bro.org/. Acedido a 17 de Novembro de 2016.

[34] "Logstash." https://www.elastic.co/products/logstash. Acedido a 18 de Novembro de 2016.

[35] "The heart of the elastic stack." https://www.elastic.co/products/elasticsearch. Acedido a 18 de Novembro de 2016.

[36] "Elasticsearch queries." https://www.elastic.co/guide/en/elasticsearch/reference/2.3/_introducing_the_query_language.html. Acedido a 22 de Novembro de 2016.

[37] "Kibana." https://www.elastic.co/products/kibana. Acedido a 18 de Novembro de 2016.

[38] "Software architecture patterns." http://www.oreilly.com/programming/free/files/software-architecture-patterns.pdf. Acedido a 20 de Dezembro de 2016.

[39] "An open-source gui prototyping tool that's available for all platforms." http://pencil.evolus.vn/. Acedido a 29 de Maio de 2017.

[40] "Debian." https://www.debian.org/index.pt.html. Acedido a 26 de Dezembro de 2016.

[41] "Kali linux - penetration testing and ethical hacking linux distribution." https://www.kali.org/. Acedido a 22 de Maio de 2017.

[42] "nginx." http://nginx.org/en/. Acedido a 11 de Maio de 2015.

[43] "netcraft." http://news.netcraft.com/archives/2015/04/20/april-2015-web-server-survey.html. Acedido a 11 de Maio de 2015.

[44] "barry wordpress." https://barry.wordpress.com/2008/04/28/load-balancer-update/. Acedido a 11 de Maio de 2015.

[45] "Will reese." http://www.linuxjournal.com/article/10108. Acedido a 11 de Maio de 2015.

[46] "Postgresql." http://www.postgresql.org/about/. Acedido a 13 de Maio de 2015.

[47] "O. s. tezer." http://tinyurl.com/o35nwah. Acedido a 13 de Maio de 2015.

[48] "Rabbitmq java client." https://github.com/rabbitmq/rabbitmq-java-client. Acedido a 16 de Maio de 2017.

[49] "Pure python rabbitmq/amqp 0-9-1 client library." https://github.com/pika/pika. Acedido a 16 de Maio de 2017.

[50] "jquery - write less, do do more." http://jquery.com/. Acedido a 4 de Maio de 2017.

[51] "Ajax (programming)." https://en.wikipedia.org/wiki/Ajax_%28programming%29. Acedido a 4 de Maio de 2017.

[52] "arbor.js a graph visualization library using web workers and jquery." http://arborjs.org/. Acedido a 4 de Maio de 2017.

[53] "Treant.js - javascipt library for visualization of tree diagrams." https://fperucic.github.io/treant-js/. Acedido a 4 de Maio de 2017.

[54] "Raphaël—javascript library." https://dmitrybaranovskiy.github.io/raphael/. Acedido a 4 de Maio de 2017.

[55] "vis.js - a dynamic, browser based visualization library." http://visjs.org/. Acedido a 4 de Maio de 2017.

[56] "json-simple." https://code.google.com/archive/p/json-simple/.

[57] "gson - a java serialization/deserialization library to convert java objects into json and back." https://github.com/google/gson. Acedido a 4 de Maio de 2017.

[58] "The ultimate json library: Json.simple vs gson vs jackson vs jsonp." http://blog.takipi.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/. Acedido a 4 de Maio de 2017.

[59] "python." https://www.python.org/about/. Acedido a 12 de Maio de 2015.

[60] "Html." http://www.w3.org/People/Raggett/book4/ch02.html. Acedido a 12 de Maio de 2015.

[61] "Css." https://www.python.org/about/quotes/. Acedido a 12 de Maio de 2015.

[62] "Javascript." https://www.javascript.com/. Acedido a 27 de Dezembro de 2016.

[63] "Json." http://www.json.org/. Acedido a 27 de Dezembro de 2016.

[64] "Gnu bash." https://www.gnu.org/software/bash/. Acedido a 27 de Dezembro de 2016.

[65] "Latex - a document preparation system." https://www.latex-project.org/. Acedido a 27 de Dezembro de 2016.

[66] "The r project for statistical computing." https://www.r-project.org/. Acedido a 27 de Dezembro de 2016.

[67] "Java." https://www.java.com/pt_BR/about/whatis_java.jsp. Acedido a 27 de Dezembro de 2016.

[68] "Django." https://www.djangoproject.com/. Acedido a 12 de Maio de 2015.

[69] "Django appears to be a mvc framework, but you call the controller the 'view' and the view the 'template'. how come you don't use standard names?." https://docs.djangoproject.com/en/1.11/faq/general/#faq-mtv. Acedido a 28 de Abril de 2017.

[70] "Bootstrap." http://getbootstrap.com/. Acedido a 19 de Maio de 2015.

[71] "Git." https://git-scm.com/. Acedido a 30 de Maio de 2017.

[72] "Intellij idea the java ide." https://www.jetbrains.com/idea/. Acedido a 18 de Maio de 2017.

[73] "Python ide for professional developers." https://www.jetbrains.com/pycharm/. Acedido a 30 de Maio de 2017.

[74] "Sublime text: The text editor you will fall in love with." http://www.sublimetext.com/. Acedido a 30 de Maio de 2017.

[75] "Gnu nano." https://www.nano-editor.org/. Acedido a 30 de Maio de 2017.

[76] "Junit." http://junit.org/junit4/. Acedido a 18 de Maio de 2017.

[77] "unittest — unit testing framework." https://docs.python.org/3/library/unittest.html. Acedido a 19 de Maio de 2017.

[78] "Graph coverage web application." https://cs.gmu.edu:8443/offutt/coverage/GraphCoverage. Acedido a 18 de Maio de 2017.

[79] "Apps/clocks - gnome wiki!." https://wiki.gnome.org/Apps/Clocks. Acedido a 31 de Maio de 2017.

[80] "Sdlc - agile model." https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm. Acedido a 03 de Fevereiro de 2017.

[81] "Sdlc - waterfall model." http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm. Acedido a 03 de Fevereiro de 2017.

[82] "Coding standards." https://ace.apache.org/docs/coding-standards.html. Acedido a 16 de Maio de 2017.

[83] "Pep 8 – style guide for python code." https://www.python.org/dev/peps/pep-0008/. Acedido a 16 de Maio de 2017.

[84] "Euclidean distance." https://en.wikipedia.org/wiki/Euclidean_distance. Acedido a 15 de Fevereiro de 2017.

[85] R. V. Hamming, "Error detecting and error correcting codes," *Bell Sys. Tech. Journal 29*, pp. 147–160, 1950.

[86] "About feature scaling and normalization." http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#standardizing-and-normalizing---how-it-can-be-done-using-scikit-learn. Acedido a 15 de Fevereiro de 2017.

[87] "User authentication in django." https://docs.djangoproject.com/en/1.10/topics/auth/. Acedido a 22 de Fevereiro de 2017.

[88] "Django url dispatcher." https://docs.djangoproject.com/en/1.11/topics/http/urls/. Acedido a 28 de Abril de 2017.

[89] "Cool uris don't change." https://www.w3.org/Provider/Style/URI. Acedido a 28 de Abril de 2017.

[90] "Design patterns - strategy pattern." http://www.tutorialspoint.com/design_pattern/strategy_pattern.htm. Acedido a 2 de Maio de 2017.

[91] "The django template language." https://docs.djangoproject.com/en/1.11/ref/templates/language/. Acedido a 3 de Maio de 2017.

[92] "React a javascript library for building user interfaces." https://facebook.github.io/react/. Acedido a 5 de Maio de 2017.

[93] "Advantages and disadvantages of using reactjs." http://stackoverflow.com/questions/28442239/advantages-and-disadvantages-of-using-reactjs. Acedido a 5 de Maio de 2017.

[94] "Verification and validation." https://en.wikipedia.org/wiki/Verification_and_validation. Acedido a 23 de Março de 2017.

[95] "Software testing - overview." http://www.tutorialspoint.com/software_testing/software_testing_overview.htm. Acedido a 23 de Março de 2017.

[96] "htop - an interactive process viewer." http://hisham.hm/htop/. Acedido a 22 de Março de 2017.

[97] "top(1) - linux man page." https://linux.die.net/man/1/top. Acedido a 22 de Março de 2017.

[98] "i* notation." http://istar.rwth-aachen.de/tiki-index.php?page=Summary%20of%20i*%20Notation&structure=i*%20Guide. Acedido a 23 de Novembro de 2016.

Part I

Apêndices

Apêndice A

Review History

Tabela A.1: Review History

Nome	Data	Razão para modificações	Versão
Daniel Margarido	17/11/2016	Criação do documento e da sua estrutura base	0.1
Daniel Margarido	18/11/2016	Escrita do Enquadramento da Solução na infraestrutura de alertas	0.2
Daniel Margarido	21/11/2016	Descrição do Enquadramento da Solução na infraestrutura de alertas, entidade acolhedora e âmbito do projeto	0.3
Daniel Margarido	22/11/2016	Escrita de contribuições, objetivos do projeto e estrutura do projeto	0.4
Daniel Margarido	23/11/2016	Escrita da escala de prioridades, estrutura dos requisitos, validação dos requisitos e entender o domínio do problema	0.5
Daniel Margarido	24/11/2016	Adição de notação i*	0.6
Daniel Margarido	25/11/2016	Escrita de Strategic Dependency Diagram, Strategic Rationale Diagram e reescrita de Entender o Domino do Problema	0.7
Daniel Margarido	28/11/2016	Escrita de Strategic Dependency Diagram, Strategic Rationale Diagram	0.8
Daniel Margarido	29/11/2016	Escrita de entender o domino do problema e atributos de qualidade	0.9

Tabela A.2: Review History II

Nome	Data	Razão para modificações	Versão
Daniel Margarido	30/11/2016	Escrita de entender o domino do problema, requisitos funcionais, conclusão da análise de requisitos e removida a validação de requisitos. Reescrita dos objectivos do projecto	0.10
Daniel Margarido	02/12/2016	Edição da estrutura do documento, escrita de fases do projeto, riscos, planeamento e conclusão do capitulo Planeamento	0.11
Daniel Margarido	05/12/2016	Escrita de riscos a abordar, etapas de projeto e planeamento	0.12
Daniel Margarido	06/12/2016	Escrita de planeamento e conclusão do planeamento	0.13
Daniel Margarido	09/12/2016	Adição de novos requisitos na análise de requisitos e escrita dos produtos no estado da arte	0.14
Daniel Margarido	12/12/2016	Adição de nome aos requisitos funcionais e não-funcionais, escrita de produtos do estado da arte e sua comparação, e escrita de descrição de papers	0.15
Daniel Margarido	13/12/2016	Escrita e adição de novos papers no capitulo do Estado da arte	0.16
Daniel Margarido	14/12/2016	Escrita e adição de novos papers no capitulo do Estado da arte	0.17
Daniel Margarido	15/12/2016	Escrita e adição de novos papers no capitulo do Estado da arte	0.18
Daniel Margarido	16/12/2016	Escrita e adição de novos papers no capitulo do Estado da arte	0.19
Daniel Margarido	19/12/2016	Escrita de papers e conclusão no capitulo do Estado da arte, edição de mockup de entender o domínio do problema e criação de mockups para arquitetura interna	0.20

Tabela A.3: Review History III

Nome	Data	Razão para modificações	Versão
Daniel Margarido	20/12/2016	Reescrita de arquitetura externa e interna do capitulo de Arquitetura do sistema	0.21
Daniel Margarido	21/12/2016	Escrita de arquitetura externa no capitulo de Arquitetura do sistema	0.22
Daniel Margarido	22/12/2016	Escrita de arquitetura interna no capitulo de Arquitetura do sistema	0.23
Daniel Margarido	26/12/2016	Escrita de arquitetura interna e ferramentas e tecnologias adotadas no capitulo de Arquitetura do sistema	0.24
Daniel Margarido	27/12/2016	Escrita de ferramentas e tecnologias adotadas no capitulo de Arquitetura do sistema, escrita de planeamento e estruturação do capitulo de Implementação	0.25
Daniel Margarido	23/01/2017	Atualização de agradecimentos, edição de ferramentas, escrita de receber logs e inicio de escrita de analisar dados no capitulo de Implementação	0.26
Daniel Margarido	24/01/2017	Atualização de agradecimentos e escrita de Statistics Based Alerts Analysis Module no capitulo de Implementação	0.27
Daniel Margarido	25/01/2017	Escrita de Máquina de estados no capitulo de implementação e pequenas correções na descrição dos alertas	0.28
Daniel Margarido	30/01/2017	Escrita de artigo sobre CBR na MyCERT no capitulo Estado Da Arte	0.29
Daniel Margarido	31/01/2017	Escrita de artigo sobre CBR na MyCERT no capitulo Estado Da Arte	0.30
Daniel Margarido	03/02/2017	Escrita de pontos a considerar e escolha de metodologia no capitulo Metodologia de Desenvolvimento	0.31

Tabela A.4: Review History IV

Nome	Data	Razão para modificações	Versão
Daniel Margarido	14/02/2016	Escrita de Classificador de resultados baseados em estatistica na seção Classificar Alertas no capitulo Implementação	0.32
Daniel Margarido	15/02/2016	Escrita de Classificador de resultados baseados em estatistica na seção Classificar Alertas no capitulo Implementação	0.33
Daniel Margarido	17/02/2016	Escrita de Classificador de resultados baseados em estatistica na seção Classificar Alertas no capitulo Implementação	0.34
Daniel Margarido	20/02/2016	Escrita de Classificador de resultados baseados em estatistica na seção Classificar Alertas no capitulo Implementação	0.35
Daniel Margarido	21/02/2016	Escrita de Classificador de resultados baseados em estatistica na seção Classificar Alertas e escrita de Investigations Database na secção Apresentar Investigações do capitulo de Implementação	0.36
Daniel Margarido	22/02/2016	Escrita de Investigations Database na secção Apresentar Investigações do capitulo de Implementação	0.37
Daniel Margarido	21/03/2016	Escrita de Persistência dos modelos de dados estatísticos na secção Statistics Based Alerts Analysis Module, atualização de Persistência dos resultados na secção Classificador de Resultados baseados em estatística e escrita e estruturação na secção Verificação e Validação	0.38
Daniel Margarido	22/03/2016	Atualização da descrição da secção de Verificação e Validação e escrita da secção de testes de performance no mesmo capitulo	0.39
Daniel Margarido	23/03/2016	Atualização da descrição da secção de Verificação e Validação, escrita da secção de testes de performance no mesmo capitulo e escrita de anexo ISTQB Tipos de Teste	0.40

Tabela A.5: Review History IV

Nome	Data	Razão para modificações	Versão
Daniel Margarido	24/03/2016	Escrita de anexo ISTQB Tipos de Teste	0.41
Daniel Margarido	27/03/2016	Fim da escrita de anexo ISTQB Tipos de Teste, atualização da secção ST07 - Perfomance no capitulo Verificação e Validação e adição de ISEC na secção de agradecimentos	0.42
Daniel Margarido	28/03/2016	Escrita de Persistência dos modelos de dados estatísticos na secção Statistics Based Alerts Analysis Module e atualização de class diagram na secção Investigations Database	0.43
Daniel Margarido	28/04/2016	Escrita de Arquitectura, ferramentas e Investigations Presentation Business Logic Module no capitulo de implementação	0.44
Daniel Margarido	02/05/2016	Escrita de Investigations Presentation Business Logic Module no capitulo de implementação	0.45
Daniel Margarido	03/05/2016	Escrita de Investigations Presentation Business Logic Module no capitulo de implementação e escrita de Waldo UI	0.46
Daniel Margarido	04/05/2016	Escrita de Waldo UI na secção de Apresentar Investigações e escrita de tratamento de eventos no cliente, representação de investigações do Waldo e Parse de alertas json na secção de Ferramentas e Tecnologias Adotadas	0.47
Daniel Margarido	05/05/2016	Escrita de Waldo UI na secção de Apresentar Investigações e atualização de diagrama de classes	0.48
Daniel Margarido	16/05/2016	Escrita de Waldo UI na secção de Apresentar Investigações, escrita de aumentar precisão de classificação, escrita de arquitetura, escrita de escolha de metodologia e inicio de ciclo de vida	0.49

Tabela A.6: Review History V

Nome	Data	Razão para modificações	Versão
Daniel Margarido	17/05/2016	Escrita de metodologia	0.50
Daniel Margarido	18/05/2016	Escrita de ferramentas para validação de funcionalidades, Secção de teste 1 e Secção de Teste 2	0.51
Daniel Margarido	19/05/2016	Escrita de ferramentas para validação de funcionalidades, Secção de teste 2 e Secção de Teste 3	0.52
Daniel Margarido	22/05/2016	Escrita de sistemas operativos em ferramentas, Secção de Teste 3	0.53
Daniel Margarido	24/05/2016	Escrita de teste de performance em testes não funcionais e escrita de automatizar tarefas no no capitulo de implementação	0.54
Daniel Margarido	25/05/2016	Escrita de adaptabilidade a várias fontes diferentes, escrita de modularidade e escrita de automatizar tarefas	0.55
Daniel Margarido	26/05/2016	Escrita de automatizar tarefas e secção de testes sobre Modularidade	0.56
Daniel Margarido	29/05/2016	Escrita de secção de testes sobre Modularidade, atualização de fonte de alertas de elasticsearch para logstash no capitulo de arquitetura, escrita de secção de teste sobre Adaptabilidade a várias fontes diferentes e descrição de ferramenta para mockups na secção das ferramentas a utilizar	0.57
Daniel Margarido	30/05/2016	Escrita de secção de testes sobre Adaptabilidade a várias fontes diferentes, descrição de ferramenta de controlo de versões e ferramentas de desenvolvimento, escrita de Redução do tempo gasto desde a identificação do alerta até à resolução do incidente	0.58
Daniel Margarido	01/06/2016	Escrita de secção de testes sobre Aumentar precisão de classificação ao longo do tempo e sobre Precisão alta de classificação de alertas	0.59

Tabela A.7: Review History V

Nome	Data	Razão para modificações	Versão
Daniel Margarido	05/06/2016	Escrita de secção de testes sobre Aumentar precisão de classificação ao longo do tempo, sobre Precisão alta de classificação de alertas e correção ortográfica do relatório	0.60
Daniel Margarido	13/06/2016	Revisão e correção geral do relatório, e escrita da redução do tempo gasto desde a identificação do alerta até à resolução do incidente s	0.61
Daniel Margarido	14/06/2016	Escrita de Precisão alta de classificação de alertas, Aumentar precisão de classificação ao longo do tempo, conclusão do capitulo de Verificação e Validação, Reflexões e Trabalho Futuro no capítulo de conclusão	0.62-rc1
Daniel Margarido	05/07/2016	Correções de sintaxe nos acrónimos, introdução, requisitos e estado da arte	0.63-rc2
Daniel Margarido	19/07/2016	Correções de resumo e acrónimos	0.64-rc2
Daniel Margarido	20/07/2016	Correções acrónimos e introdução	0.65-rc2
Daniel Margarido	25/07/2016	Correções na introdução	0.66-rc2
Daniel Margarido	26/07/2016	Correções na introdução, adição de relatório de estágio nos apêndices e descrição do Strategic Dependency Model na Análise de Requisitos.	0.67-rc2
Daniel Margarido	27/07/2016	Descrição do Strategic Dependency Model, Strategic Rationale Model e tabelas na Análise de Requisitos.	0.68-rc2
Daniel Margarido	03/08/2016	Strategic Rationale Model, tabelas na Análise de Requisitos e correção geral em vários dos pontos sobre requisitos.	0.69-rc2
Daniel Margarido	04/08/2016	Correções e simplificação da explicação dos artigos no capitulo do Estado da arte.	0.70-rc2
Daniel Margarido	05/08/2016	Correções e simplificação da explicação dos artigos e da conclusão no capitulo do Estado da arte.	0.71-rc2

Tabela A.8: Review History V

Nome	Data	Razão para modificações	Versão
Daniel Margarido	06/08/2016	Correções e simplificação da explicação dos artigos e da conclusão no capitulo do Estado da arte, correções no capitulo de planeamento, correções e simplificação de arquitectura e ferramentas no capitulo de arquitectura do sistema.	0.72-rc2
Daniel Margarido	07/08/2016	Correção e explicação no capitulo de metodologia de desenvolvimento e correções nas secções 7.1, 7.2 e 7.3 do capitulo de implementação.	0.73-rc2
Daniel Margarido	08/08/2016	Correções nas secções 7.3, 7.4, 7.5 e 7.6 do capitulo de implementação.	0.74-rc2
Daniel Margarido	09/08/2016	Correções nas secções do capitulo de implementação, correções no capitulo de validação, escrita de trabalho realizado na conclusão e correções na bibliografia.	0.75-rc2
Daniel Margarido	12/09/2016	Correções ortográficas globais, corrigidos pontos das funcionalidades do Waldo na tabela comparativa no Estado da Arte, removida capa antiga e troca de folha de rosto pela pré-definida pelo ISEC.	0.76-rc3
Daniel Margarido	13/09/2016	Correções ortográficas globais, adição de abstract junto ao resumo e explicação mais detalhada do mesmo. Colocados inicios de capítulos em páginas impares.	1.0

Apêndice B

i* Notation

Para permitir uma mais fácil leitura dos modelos i*, este capítulo apresenta todos os modelos da notação utilizada[98].

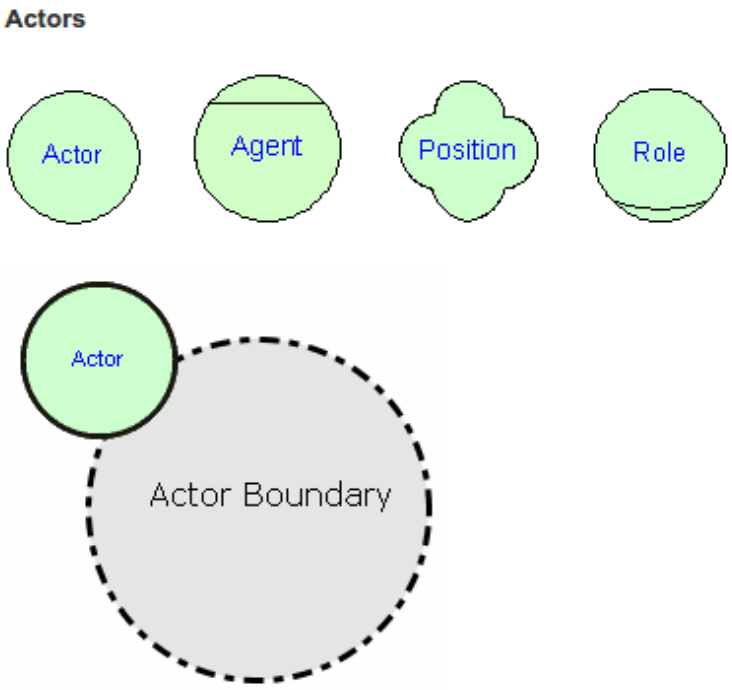


Figura B.1: i* Actors

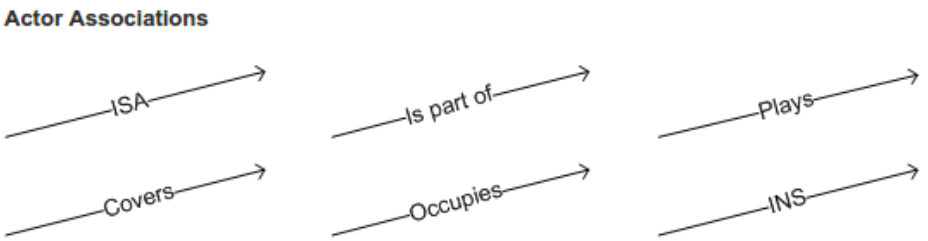


Figura B.2: i* Actors Associations

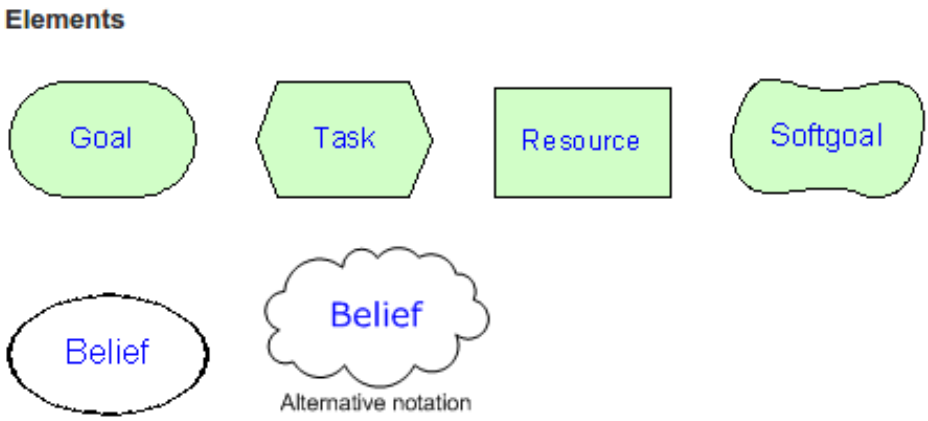


Figura B.3: i* Elements

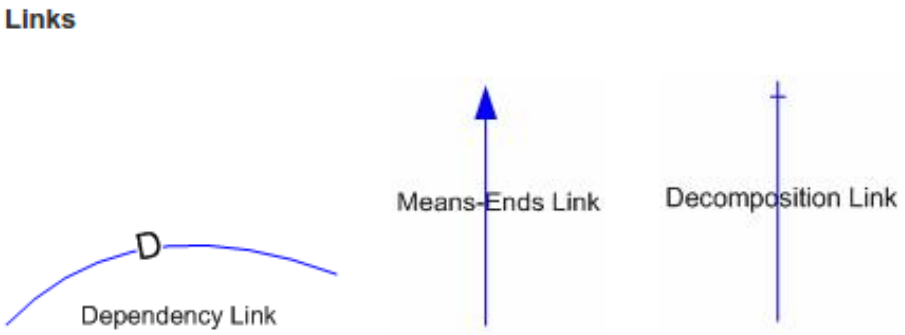


Figura B.4: i* Links

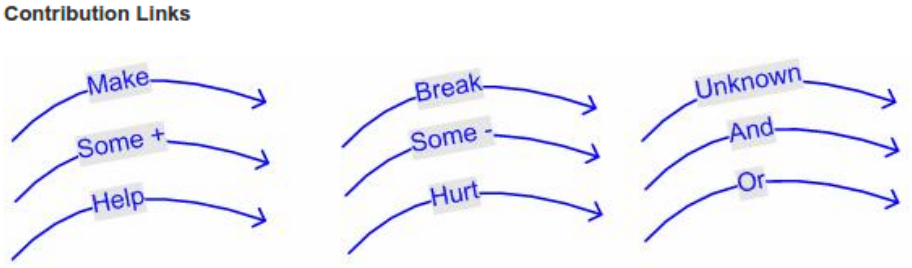


Figura B.5: i* Contribution Links

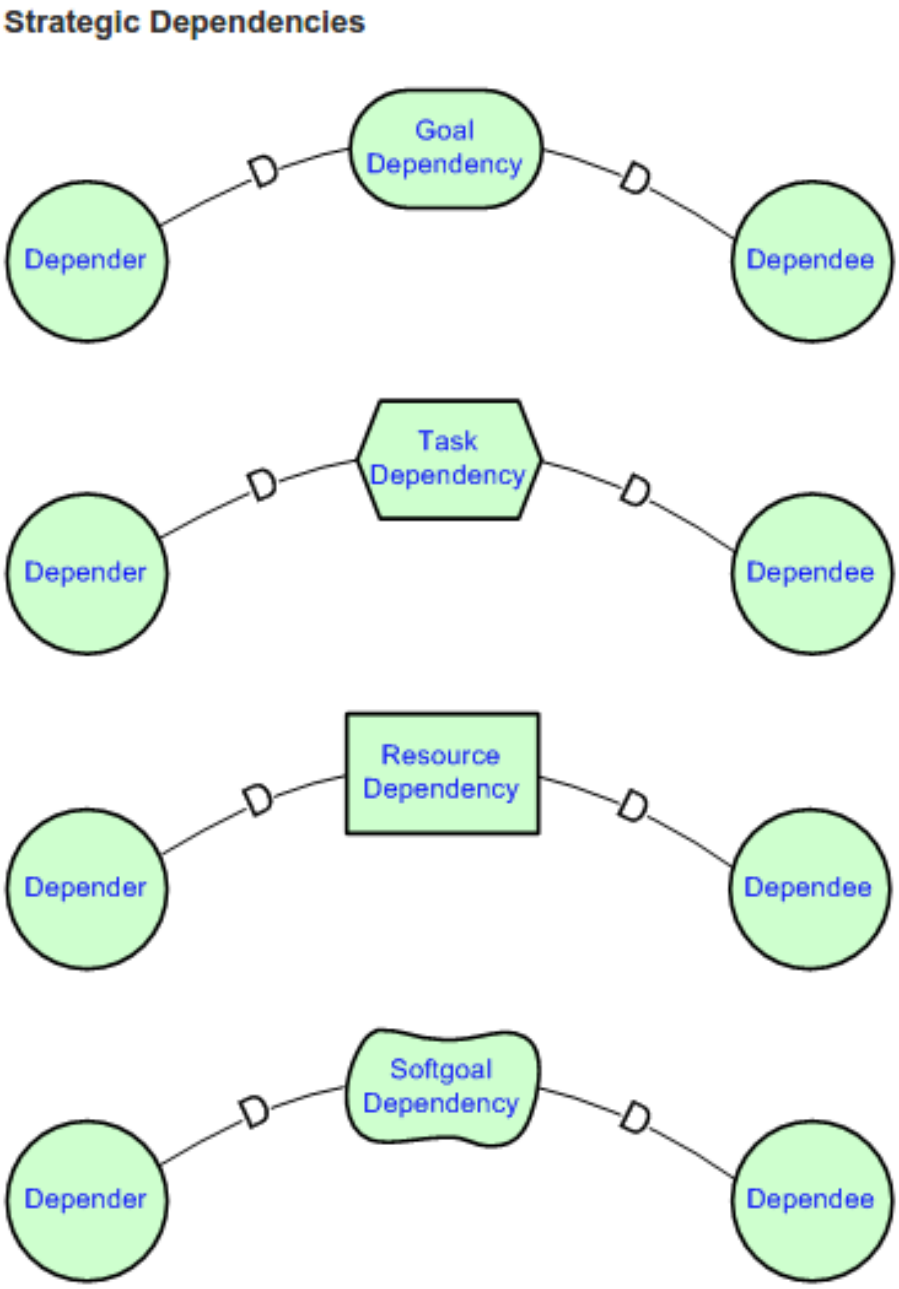


Figura B.6: i* Strategic Dependencies

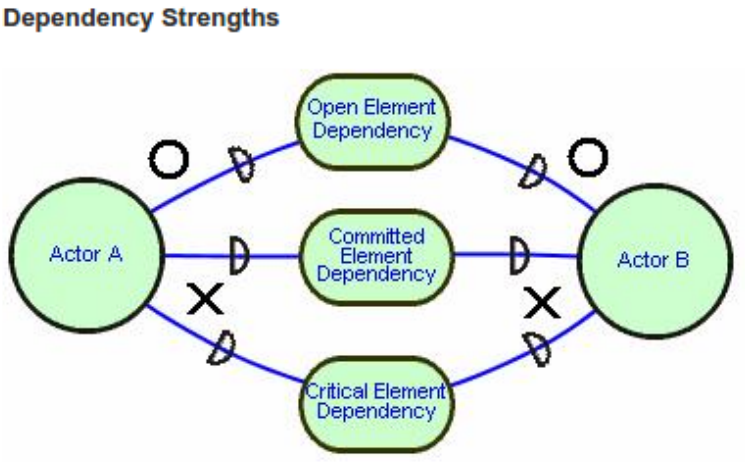


Figura B.7: i* Dependendy Strengths

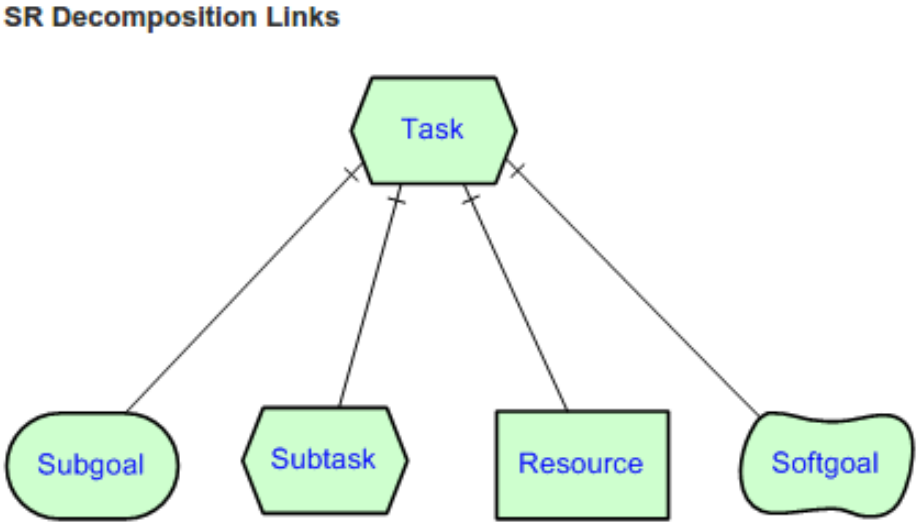


Figura B.8: i* SR Decomposition Links

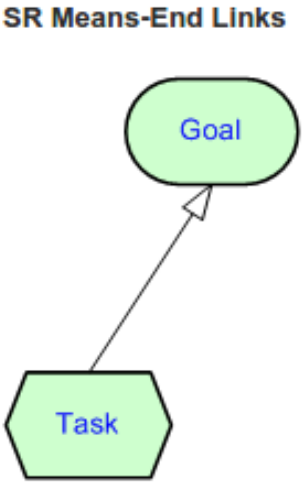


Figura B.9: i* SR Mean-End Links

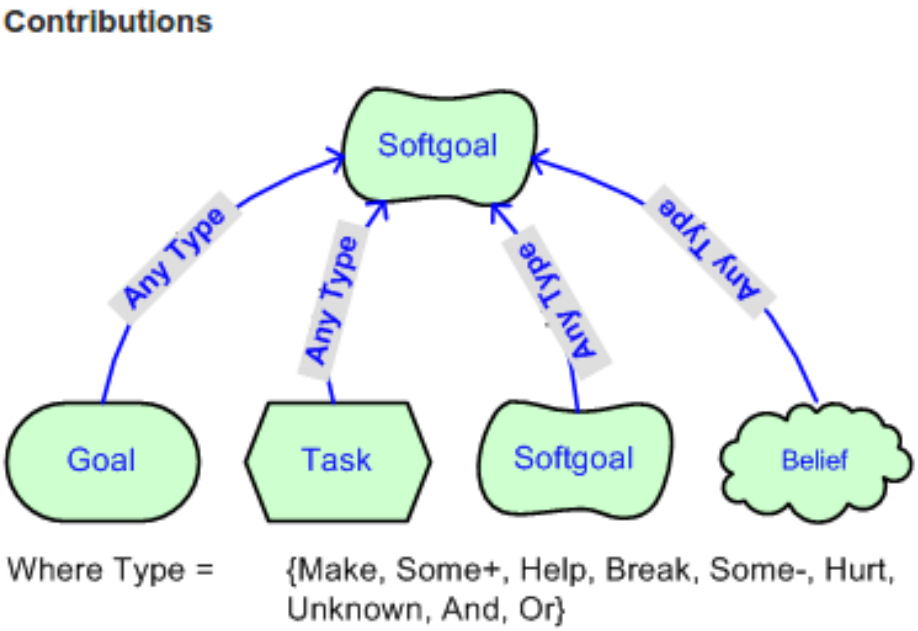


Figura B.10: i* Contributions

Apêndice C

ISTQB Tipos de Teste

Este apêndice apresenta algumas bases sobre testes de software e explica os vários tipos de teste consoante as descrições apresentadas no Foundation Level Syllabus[10] da ISTQB.

C.1 Niveis de Teste

Os testes podem ser realizados em diferentes níveis do sistema, estes níveis dividem-se em:

- **Component(Unit) testing**:
  - Procura por defeitos e valida o funcionamento de módulos de software, programas, objetos, classes, entre outros que sejam isoladamente testáveis;
  - Pode testar características funcionais e não funcionais do sistema;
  - Regularmente é realizado com acesso ao código que está a ser testado, com suporte de uma framework de testes unitários e pelo próprio programador que escreveu o código;
- **Integration testing**:
  - Testa as interfaces e interações entre componentes com diferentes partes de um sistema, como um sistema operativo, um sistema de ficheiros, o hardware e as interfaces entre estes;
  - A cada etapa de integração, os testers concentram-se apenas na integração em si. Por exemplo se forem integrados dois módulos o que é testado é a comunicação entre estes e não as funcionalidades individuais de cada um;
- **System testing**:

  - Preocupa-se com o comportamento de produto/sistema como um todo;
  - O ambiente de teste deve corresponder ao ambiente de produção para minimizar o risco de certas falhas especificas ao ambiente de produção não serem encontradas durante a fase de testes;
  - Inclui testes baseado nos riscos, na especificação de requisitos, processos de negócio, casos de uso, ou outras descrições de alto nível de modelos ou comportamentos do sistema.

- **Acceptance testing**:
  - São feitos normalmente pelo cliente ou utilizadores do sistema, podendo também estar envolvidos outros stakeholders;
  - O objetivo principal na acceptance testing não é encontrar defeitos, é estabelecer confiança no sistema, partes do sistema ou caraterísticas não funcionais do sistema;

C.2 Testing Techniques

Nesta secção são então apresentadas várias técnicas de teste, sendo que uma técnica pode estar relacionada a vários dos tipos apresentados, contudo vão aparecer no que estão mais próximas. Não estão aqui apresentadas todas as técnicas existentes, apenas uma síntese das técnicas usadas neste estágio e algumas próximas a estas ou que se considerem relevantes apresentar para complementar o conhecimento base de testes de software.

C.2.1 Static Techniques

Tecnicas estáticas realizam examinarão e inspeção do código e da documentação do projeto sem executar código.

Review

Defeitos detetados durante as reviews no inicio do ciclo de vida de desenvolvimento do sistema são muito mais baratos de remover do que os que forem detetados ao correr os testes executando o código. As reviews tem como benefícios:

- Deteção de defeitos cedo no projeto e sua correção;
- Aumento de produtividade no desenvolvimento;
- Reduzido tempo de teste;
- Menos defeitos;

- Comunicação melhorada;
- As reviews podem encontrar omissões, por exemplo, de requisitos que são pouco prováveis de ser encontradas nos testes dinâmicos(testes que executam código).

Consoante a sua formalidade estas dividem-se em:

- **Informal Review** - É informal, pode tomar a forma de pair programming ou o líder técnico rever o código e o design. Tem como principal objetivo obter algum beneficio de forma barata;
- **Walkthrough** - Reunião dirigida pelo autor, pode tomar o formato de cenários, testes simples com a participação de equipa técnica. Possui opcionalmente um escritor(que não seja o autor), pode variar desde bastante informal para bastante formal. Tem como principais objetivos aprendizagem, permitir à equipa compreender e encontrar defeitos;
- **Technical Review** - É documentada, utiliza algum processo de registo de deteção de defeitos e conta com a participação de membros técnicos e opcionalmente da gestão. Deve ser dirigida por um moderador treinado(não pode ser o autor), exige alguma preparação para o meeting por todos os reviewers, tem como output um relatório de review que inclui a lista de defeitos encontrados, o veredicto de que o produto cumpre com os requisitos e recomendações relacionadas aos defeitos encontrados. Tem como objetivo discussão, a tomada de decisões, avaliar alternativas, encontrar defeitos, resolver problemas técnicos, e validar conformidade com especificações, planos regulamentos e standards;
- **Inspection** - Liderada por um moderador treinado(não pode ser o autor), existem papeis definidos, é um processo formal baseado em regras e checklists. Possui um critério de aceitação já definido para o produto, é necessária preparação para a inspeção por parte dos vários intervenientes, opcionalmente pode ser atribuído o role de leitor , tem como resultado a lista de defeitos encontrados e como objetivo encontrar defeitos.

Static Analysis by Tools

O objetivo da análise estática é encontrar defeitos no código fonte e modelos de software. Esta análise tem como principal valor:

- Deteção de defeitos cedo antes da execução de testes;
- Avisos de aspetos suspeitos do código ou design através do cálculo de métricas;

- Identificação de defeitos não encontrados facilmente por testes dinâmicos;
- Detetar dependências e inconsistências em modelos de software;
- Melhoramento na facilidade de manutenção do código e design;
- Prevenção de defeitos caso as lições sejam aprendidas no desenvolvimento.

C.2.2 Black-box Techniques

As técnicas baseadas em especificação(Black-box) tem como principais caraterísticas:

- Modelos, tanto formais como informais, que são usados para a especificação do problema a ser resolvido, do software ou dos seus componentes;
- A partir destes modelos podem ser derivados vários casos de teste.

Equivalence Partitioning

Na Equivalence Partitioning, os inputs para o software ou sistema são divididos em grupos que são expectáveis de exibir comportamentos semelhantes, de modo a poderem ser processados da mesma forma. Equivalence Partitions(ou classes) podem ser encontradas para:

- Outputs;
- Valores internos;
- Valores temporais;
- Parâmetros de interface;
- Dados válidos(por exemplo valores que devem ser aceites);
- Dados inválidos(por exemplo valores que devem ser rejeitados);

Os testes podem ser desenhados para cobrir todas as partições válidas e inválidas. Este tipo de abordagem pode ser utilizada em todos os níveis de teste, para atingir objetivos de coverage de inputs e outputs, para input humano, para input via interfaces para um sistema ou parâmetros de interface em testes de integração.

Boundary Value Analysis

Os comportamentos nos limites de cada Equivalence Partition são mais prováveis de estar incorretos que os comportamentos dentro da partição, deste modo os limites são áreas em que os testes tem uma maior probabilidade de encontrar defeitos. Para determinar os limites e testes a partir destes tem-se em conta que:

- O valor máximo e mínimo de uma partição são os seus valores limite;
- O valor de limite para uma partição válida é um valor válido da partição;
- O limite de uma partição inválida é um valor limite invalido;
- Testes podem ser desenhados para cobrir ambos os valores limite, válidos e inválidos;
- Deve ser escolhido um teste para cada valor limite.

A técnica Boundary Value Analysis apresenta as seguintes características:

- Pode ser aplicada a todos os níveis de teste;
- É relativamente simples de aplicar e tem uma grande capacidade de encontrar defeitos;
- Especificações detalhadas são úteis para determinar limites interessantes;
- É frequentemente considerada uma extensão da Equivalence Partitioning ou outras técnicas black-box de design de testes.

Decision Table Testing

Tabelas de decisão são uma boa forma de:

- Encontrar requisitos de sistema que contenham condições lógicas;
- Documentar o design interno do sistema;
- Gravar regras de negócio complexas que um sistema pretende implementar;
- Criar combinações de condições que de outra forma poderiam não ser exercitadas durante os testes.

Para contruir uma tabela de decisão:

- A especificação é analisada e são identificadas condições e ações do sistema;

- As condições e ações de input são mais frequentemente descritas de forma a que estas sejam verdadeiras ou falsas(Boolean);
- A tabela de decisão contêm as condições inicias, frequentemente combinações de verdadeiros e falsos para todas as condições de input, e as ações resultantes para cada combinação de condições;
- Cada coluna da tabela corresponde a uma regra de negócio que define uma combinação única de condições e que resulta na execução de ações associadas a essa regra.

A quantidade comum de coverage utilizada nestas tabelas de decisão é regularmente ter um teste por coluna na tabela, o que envolve cobrir todas as condições iniciais. Pode ser aplicada a todas as situações que a ação do software dependa de decisões lógicas.

State Transition Testing

Um sistema pode apresentar uma resposta diferente dependendo das suas condições atuais e consoante o seu histórico(o seu estado). Neste caso, esse aspeto do sistema pode ser apresentado com um diagrama de transição de estados. Este permite ao tester:

- Visualizar o software em termos de estados;
- Transições entre estados;
- Inputs ou eventos que despoletem a mudança de estado(transições);
- Ações que podem resultar das transições;
- Obter os estados de modo separado, identificado e num numero finito;
- Construir uma tabela de estados que apresenta os relacionamentos entre estados e inputs, podendo destacar possíveis transições que sejam inválidas.

Testes podem ser desenhados para cobrir uma sequência de testes, para cobrir todos os estados, para exercitar todas as transições, para exercitar sequências especificas de transições ou para testar transições inválidas. State Transition Testing é bastante utilizada na indústria de *embedded software* e automação técnica. No entanto, esta técnica é apropriada para modelar um objeto de negócio tendo estados especificos ou para testar um fluxo de janelas de diálogo.

Use Case Testing

Os casos de uso tem as seguintes caraterísticas:

- Descrevem interações entre atores(utilizadores ou sistemas), produzindo valor para o utilizador do sistema ou o cliente;
- Podem ser descritos num nível abstrato(caso de uso de negócio, sem tecnologia, nível de processo de negócio) ou ao nivel do sistema(caso de uso do sistema sobre o nível de funcionalidade);
- Tem pré-condições que precisam ser cumpridas para que o este funcione corretamente;
- Termina com pós-condições que são os resultados observáveis e o estado final do sistema após o caso de uso ter sido completo;
- Tem um cenário principal e cenários alternativos;

Deste modo entende-se que os casos de uso descrevem um fluxo de processos num sistema baseado na forma como vai ser utilizado, então os casos de teste derivados destes são úteis para:

- Encontrar defeitos no fluxo de processos que será feito no sistema num ambiente real;
- Para o design de testes de aceitação com participação do cliente;
- Encontrar defeitos de integração e interferência dos vários componentes, que os testes individuais aos componentes não conseguem ver.

Ao desenhar casos de teste a partir dos casos de uso podemos combinar estes com outras técnicas baseadas em especificação(Black-box).

C.2.3 White-box Techniques

As técnicas baseadas em estrutura(White-box) tem como principais caraterísticas:

- Informação de como o software é construido, sendo esta usada para criar casos de teste(por exemplo, código e informação detalhada de design);
- A coverage do software pode ser medida para os casos de teste existentes e futuros casos de teste podem ser derivados para aumentar a coverage.

Statement Testing and Coverage

No teste de componentes, a statement coverage é a avaliação da percentagem de linhas que foram exercitadas pelos testes criados sobre o numero total de linhas executáveis de código a ser testado. A técnica de statement testing cria casos de teste que executam linhas especificas, normalmente para aumentar a statement coverage.

Decision Testing and Coverage

Decision coverage (ou branch coverage) é a avaliação da percentagem dos resultados de decisões(opções true e false de uma linha if) que foram exercitadas por um conjunto de testes. Decision coverage é mais forte que statement coverage, 100% de decision coverage assegura 100% statement coverage, mas não vice-versa.
A técnica de decision testing cria casos de teste para executar resultados de decisões especificas, controlando o fluxo de cada teste através dos vários pontos de decisão. Ramos tem origem nos pontos de decisão no código e demonstram a transferência de fluxo para diferentes sítios no código.

C.2.4 Experience-Based Techniques

Os testes baseados em experiência tem como principais características:

- Utilizam o conhecimento, intuição e experiência do tester com aplicações e tecnologias parecidas para derivar casos de teste;
- Tiram proveito do conhecimento dos testers, developers, utilizadores e outros stakeholders sobre o software, a sua utilização e o seu ambiente como fontes de informação;
- Aproveitam conhecimento sobre defeitos prováveis e a sua distribuição como outra fonte de informação;
- Podem ser úteis na identificação de testes especiais que dificilmente seriam encontrados com as técnicas formais, especialmente quando aplicadas após as técnicas formais.

Uma técnica baseada em experiência que é frequentemente utilizada é o *error guessing*. Genericamente os testers antecipam defeitos baseando-se na sua experiência. Uma abordagem estruturada para a técnica de *error guessing* é enumerar uma lista de possíveis defeitos e desenhar testes que ataquem esses defeitos. Esta abordagem é chamada de *fault attack*.
Testes exploratórios utilizam como base uma tabela contendo os objetivos de teste e estão limitados a um certo tempo limite em que se tem de realizar simultaneamente:

- Design de testes;
- Execução de testes;
- Logging dos testes;
- Aprendizagem.

É uma abordagem que é mais útil quando há poucas ou inadequadas especificações e severa pressão temporal, ou para complementar outra abordagem mais formal de teste. Pode servir como validação no processo de teste de modo a validar que os defeitos mais sérios foram encontrados.

Apêndice D

Proposta de Estágio

Aqui é apresentada a proposta de estágio recebida pela organização.

DEPARTAMENTO DE ENGENHARIA
INFORMÁTICA E DE SISTEMAS

PROPOSTA DE ESTÁGIO

Ano Lectivo de 2016/2017

em Mestrado em Informática e Sistemas (Desenvolvimento de Software)

TEMA

Waldo, the Virtual & Intelligent Cyber Analyst

SUMÁRIO

Este estágio propõe o desenvolvimento de uma plataforma inteligente, que se pretende que constitua um analista de segurança de informação virtual, capaz de inferir possíveis incidentes de segurança, através da agregação e correlação de várias fontes, apresentando-as ao operacional de segurança sobre uma aplicação WEB.

Para tal, esta deverá aferir e registar o comportamento de operadores humanos, em trabalho de monitorização de segurança de redes de comunicação, aprendendo com estes através de modelos de *Machine Learning*. Tal facilitará a posterior interação dos operadores, com os sistemas de detecção, prevenção e registo de possíveis intrusões na rede e serviços.

A plataforma deverá ser capaz de fazer a correlação de dados de diferentes plataformas e registos, evoluindo conforme o feedback que vá sendo dado pelo operador, às correlações de informação apresentadas pela plataforma.

Este é um trabalho com uma forte componente de investigação na área de Inteligência Artificial, visto que se pretende que o desenvolvimento de algoritmos de recolha e tratamento de dados, leve a que a máquina seja 'treinada' com os dados fornecidos, para passar a executar determinadas tarefas em eventos futuros.

1. ÂMBITO

A Dognaedis é uma empresa com foco em questões de segurança de informação, fornecendo aos seus clientes, entre outros serviços, assentes sobre outras áreas de negócio, a monitorização continua e resposta a incidentes que possam incidir sobre as suas redes de comunicação e serviços de informação.

Para tal, a Dognaedis faz-se valer de um conjunto de plataformas de monitorização e de consolas agregadoras de registos escrito por terminais e equipamentos de rede, cabendo actualmente ao analista de segurança, correlacionar informação patente nestes registos e plataformas, para detectar possíveis incidentes de segurança, que como tal devam ser tratados e mitigados.

Com a implementação da plataforma de aprendizagem aqui proposta, o trabalho manual de correlação de eventos seria amplamente simplificado, levando a uma melhoria não só na eficiência do operador de segurança na reação a possíveis eventos de intrusão, mas também numa aprendizagem continua entre operador e plataforma, na pertinência que determinados registos possam ter para a identificação de incidentes.

2. OBJECTIVOS

O estágio proposto, tendo uma forte componente de investigação e sendo algo inovador na área em que se insere, pretende mudar o paradigma de análise e correlação de registos. Ao invés do operador fazer pesquisas manuais sobre extensas listas de registos de rede e máquina, passará a moldar o comportamento da plataforma, através de respostas a sugestões apresentadas por esta, dados comportamentos seus (ou de outro operador), em incidentes que se tenham verificado anteriormente.

O presente projecto pretende atingir os seguintes objectivos genéricos :

- Implementar um mecanismo de aprendizagem através do qual a plataforma detecte e registe os comportamentos dos operadores de monitorização, no seu trabalho do dia-a-dia;
- Aprender com tais comportamentos, levando a que caso a mesma situação se repita, sugira ao operador o conjunto de comportamentos que tenham sido tomados anteriormente;
- Permita ao operador, ir moldando a aprendizagem da plataforma, enviando feedback se determinada informação correlacionada, é ou não util para a detecção, ou seguimento de determinado incidente.
- Com base nos registos tratados e em comportamentos anteriores, possa sugerir nova informação ao operador, que apesar de não ter sido utilizada anteriormente por este, se possa mostrar relevante no âmbito da análise em que se encontra.

A linguagem de programação e tecnologias a utilizar para a elaboração da plataforma são do critério do estagiário. Não obstante, dadas as suas funcionalidades da plataforma, será necessário a integração com as plataformas da empresa desenvolvidas em Python, mesmo que por intermédio de uma API REST. A utilização de tecnologias como HTML, JS e CSS serão sempre necessárias para apresentação de resultados ao utilizador.

3. PROGRAMA DE TRABALHOS

O estágio consistirá nas seguintes actividades e respectivas tarefas:

- ***T1*** – *Estudo de Plataformas* – Nesta fase o estagiário perceber a forma de operação das plataformas de monitorização de rede e a informação que estas fornecem ao operador de monitorização para a realização do seu trabalho.
- ***T2*** – *Levantamento de Requisitos* – Dadas as plataformas e registos existentes, o estagiário deverá em T2, perceber quais os requisitos necessários ao registo de informação e comportamento, correlação de eventos e apresentação de informação ao operador, bem como recepção e tratamento de feedback. Requisitos Funcionais, Restrições,
- ***T3*** – *Desenvolvimento* – Desenvolvimento dos requisitos identificados em T2.
- **T4** – *Testes* – Testes funcionais e atributos de qualidade ao trabalho desenvolvido ao longo das tarefas anteriores.
- **T5** – *Relatório* – Relatório de Estágio.

4. CALENDARIZAÇÃO DAS TAREFAS

As Tarefas acima descritas, incluindo os testes de validação de cada módulo, serão executadas de acordo com a seguinte calendarização:

O plano de escalonamento dos trabalhos é apresentado em seguida, sendo que está planeada a execução de 1200 horas de trabalho, o que equivale a 30 semanas.

INI		**Início dos trabalhos**
M1	(INI + 1 Semanas)	Tarefa T1 terminada
M1	(INI + 4 Semanas)	Tarefa T2 terminada
M2	(INI + 16 Semanas)	Tarefa T3 terminada
M5	(INI + 4 Semanas)	Tarefa T4 terminada
M7	(INI + 6 Semanas)	Tarefa T5 terminada

5. RESULTADOS

Os resultados a apresentar pelo estagiário serão consubstanciados nas várias metas estabelecidas para o projecto, sendo que serão espectáveis 2 períodos de avaliação fundamentais:

Dezembro de 2016

- Avaliação preliminar de arquitectura: Serão analisados os resultados do trabalho efectuado para as tarefas T1 e T2, verificando a exequabilidade da arquitectura e modelo de aprendizagem propostos;

Julho de 2017

- Avaliação da plataforma e de resultados: Será avaliada nesta fase a adequabilidade e resposta da proposta desenvolvida aos requisitos propostos inicialmente pela Dognaedis;
- Será também avaliada a qualidade da documentação elaborada pelo estagiário, materializada no seu relatório de estágio.

6. LOCAL DE TRABALHO

O trabalho será realizado apartir dos escritórEnigmailios da Dognaedis em Coimbra.

Horário de Trabalho: 9h00 às 18h00, com uma hora para almoço.

7. METODOLOGIA

Embora estejam salvaguardadas 4 semanas no final do projecto, o dossier de projecto deverá ser elaborado pelo estagiário ao longo da execução do projecto, com o apoio do orientador.

Prevê-se que o desenvolvimento seja assente numa metodologia ágil, assegurando a realização de reuniões semanais de acompanhamento ao projecto, sobre as quais será verificada a conformidade com o planeamento estabelecido neste documento enquanto macro-tarefas e com o planeamento delineado pelo estagiário para execução da fase de implementação.

8. ORIENTAÇÃO

ISEC:

Nome (nome@isec.pt)
Categoria

Entidade de Acolhimento:

Nome (ramaro@Dognaedis.com)
Cargo Service Line Manager, QISMS Manager